NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IMPROVING AUTOMATED LEXICAL AND DISCOURSE ANALYSIS OF ONLINE CHAT DIALOG

by

Eric Nielsen Forsyth

September 2007

Thesis Advisor:	Craig H. Martell
Second Reader:	Kevin M. Squire

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2007	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Improving Automated Lexical and Discourse Analysis of Online Chat Dialog		**5. FUNDING NUMBERS**	
6. AUTHOR(S) Eric Nielsen Forsyth			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		**8. PERFORMING ORGANIZATION REPORT NUMBER**	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited		**12b. DISTRIBUTION CODE**	

13. ABSTRACT (maximum 200 words)

One of the goals of natural language processing (NLP) systems is determining the meaning of what is being transmitted. Although much work has been accomplished in traditional written and spoken language domains, little has been performed in the newer computer-mediated communication domain enabled by the Internet, to include text-based chat. This is due in part to the fact that there are no annotated chat corpora available to the broader research community. The purpose of our research is to build a chat corpus, initially tagged with lexical and discourse information. Such a corpus could be used to develop stochastic NLP applications that perform tasks such as conversation thread topic detection, author profiling, entity identification, and social network analysis.

During the course of our research, we preserved 477,835 chat posts and associated user profiles in an XML format for future investigation. We privacy-masked 10,567 of those posts and part-of-speech tagged a total of 45,068 tokens. Using the Penn Treebank and annotated chat data, we achieved part-of-speech tagging accuracy of 90.8%. We also annotated each of the privacy-masked corpus's 10,567 posts with a chat dialog act. Using a neural network with 23 input features, we achieved 83.2% dialog act classification accuracy.

14. SUBJECT TERMS Computer-Mediated Communication, Chat, Natural Language Processing, Part-of-Speech Tagging, Discourse, Dialog Act			**15. NUMBER OF PAGES** 127
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

IMPROVING AUTOMATED LEXICAL AND DISCOURSE ANALYSIS OF ONLINE CHAT DIALOG

Eric N. Forsyth
Major, United States Air Force
B.S., University of Michigan, 1991
M.S., Purdue University, 1993

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2007

Author: Eric N. Forsyth

Approved by: Craig H. Martell, PhD
Thesis Advisor

Kevin M. Squire, PhD
Second Reader

Peter J. Denning, PhD
Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

One of the goals of natural language processing (NLP) systems is determining the meaning of what is being transmitted. Although much work has been accomplished in traditional written and spoken language domains, little has been performed in the newer computer-mediated communication domain enabled by the Internet, to include text-based chat. This is due in part to the fact that there are no annotated chat corpora available to the broader research community. The purpose of our research is to build a chat corpus, initially tagged with lexical and discourse information. Such a corpus could be used to develop stochastic NLP applications that perform tasks such as conversation thread topic detection, author profiling, entity identification, and social network analysis.

During the course of our research, we preserved 477,835 chat posts and associated user profiles in an XML format for future investigation. We privacy-masked 10,567 of those posts and part-of-speech tagged a total of 45,068 tokens. Using the Penn Treebank and annotated chat data, we achieved part-of-speech tagging accuracy of 90.8%. We also annotated each of the privacy-masked corpus's 10,567 posts with a chat dialog act. Using a neural network with 23 input features, we achieved 83.2% dialog act classification accuracy.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The research presented in this thesis is due to the varied contributions of many people, so it is only proper that I recognize the significance of their support.

Dr. Craig Martell, as my thesis advisor you gave me something that many in your position do not afford to their students: the freedom to figure out that which truly interested me. Thank you for providing me sage advice along the road to discovery.

Dr. Kevin Squire, you gave me a great appreciation for a glamourless aspect of computer science that I somehow missed in all my years of engineering programming: the data structure. Much of my research relied on those linked lists, trees, and hash tables that never seem to get their due.

Ms. Jane Lin, if you hadn't shared your interest in computer-mediated communication, I would still be writing my thesis proposal. Sometimes the hardest part of great research is getting the data. It is my hope that your data will form a foundation for tomorrow's discoveries in our field.

To my fellow computer science students, I have been so impressed not only with your intellect, but also your selfless service to country. Thank you for your words of advice and encouragement. I wish you the very best in your military careers and beyond.

Words are not adequate to express the love and support of my family. Mom and Dad, you have been the prototypical cheerleaders that all great parents are for their kids. Greg, as my brother and dear friend, your no-nonsense approach to life has always set an example for me; I aspire to be in your league.

Most important, though, are those who stand with me every day. Kate, I am so excited that your formal journey of learning has begun with kindergarten this year; you are such as bright, beautiful girl. Your dry sense of humor always lifts me up. Maxwell, you are so wise beyond your years; I often forget that you are an eight-year old still finding his way. Your love for music, friendship, and life inspire all those lucky enough to know you. Finally, it is not easy to pack up your life every two years, leaving behind home and friendships. Michelle, you get us through those moves; you provide the family with stability while I am away; you encourage our children to be their very best; and through your actions, you impress upon us that it is not what we say but rather what we do that counts. Although I'll never be a quilter of your stature, hopefully some tennis lessons will help make me a worthy competitor someday! I love you dearly.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Computer-mediated communication (CMC), as defined by Herring, is "communication that takes place between human beings via the instrumentality of computers [1]." Per this definition, the CMC domain, which is distinct from traditional written and spoken domains, includes genres such as e-mail, newsgroups, weblogs, instant messaging (IM), and text-based chat.

Chat is distinguished from the other CMC genres based on the "near-synchronous" participation of multiple users spatially separated from one another. This seemingly simple concept, powered by the Internet, has permitted groups of people to not only communicate with one another, but to collaborate real-time on problems they collectively face. Indeed, a perfect example of this is military use of text-based chat, which has supplanted traditional command and control (C2) systems as a primary way of moving time-critical information around the tactical environment [2].

Orthogonal to written, spoken, and CMC domains is the development of natural language processing (NLP) applications to enhance communication itself. Many examples exist where NLP applications tailored for the written and spoken domains are changing the way we live. These include spelling- and grammar-checking on our word processing software; voice-recognition in our automobiles; and telephone-based conversational agents that help us troubleshoot our personal and business account issues. Even more sophisticated "semantic" applications are currently under development, such as automated tools that assist in the identification of entities in written (electronic) documents along with the associated social networks that tie those entities together.

Chat, as an example of the CMC domain, can also benefit from NLP support. For instance, text-based conversational agents can help customers make purchases on-line [3]. In addition, discourse analyzers can automatically separate multiple, interleaved conversation threads from chat rooms either in real-time or after the fact in support of

information retrieval applications. Finally, author-profiling tools can help detect predatory behavior in a recreational chat setting, or even the illegitimate use of chat by terrorist and other criminal organizations.

Most NLP applications are stochastic in nature, and are thus trained on corpora, or very large samples of language usage, tagged with lexical, syntactic, and semantic information. The Linguistic Data Consortium (LDC), an open organization consisting of universities, companies, and government research laboratories, was founded in 1992 to help create, collect, and distribute such databases, lexicons, and other resources for computer-based linguistic research and development. As of August 2007, the LDC has made available 381 text-, audio-, and video-based corpora to the larger research community [4].

Not surprisingly, the effectiveness of an NLP application for a particular domain is largely influenced by the information it learns during training. As noted by the LDC,

> Different sorts of text have different statistical properties—a model trained on the Wall Street Journal will not do a very good job on a radiologist's dictation, a computer repair manual, or a pilot's requests for weather updates…This variation according to style, topic and application means that different applications benefit from models based on appropriately different data—thus there is a need for large amounts of material in a variety of styles on a variety of topics—and for research on how best to adapt such models to a new domain with as little new data as possible [4].

However, of the 381 corpora provided by the LDC, only three contain samples from the CMC domain and none from chat in particular. And yet, CMC and chat are not going away anytime soon.

Thus, as noted earlier by LDC, if we seek to build NLP applications for chat, we must accomplish two things: 1) Collect chat data and annotate it with lexical, syntactic, and semantic information; and 2) Adapt existing resources (both corpora from other domains and NLP algorithms) in conjunction with this annotated chat corpus to tailor automated tools to support chat use. These two observations form the foundation of our research presented in this thesis.

B. ORGANIZATION OF THESIS

We have organized this thesis as follows. In Chapter I we provide a motivation for the creation of an online chat corpus with tailored NLP techniques. In Chapter II we provide a synopsis of previous work in the area, to include: 1) The linguistic study of chat and comparison to traditional spoken and written communication domains; 2) How chat is currently used today, and where it can benefit from NLP; and 3) A review of general NLP techniques we will bring to bear on our research, to include annotated corpora, part-of-speech tagging, and dialog act modeling. In Chapter III we detail our technical approach, to include: 1) The approach we used to build the chat corpus; 2) The supporting mathematical foundation for the algorithms we used in both automated part-of-speech tagging and chat dialog act classification; and 3) The experimental set-up we used to test the effectiveness of those algorithms. In Chapter IV we discuss our results, to include: 1) The lexical statistics we collected from our chat corpus, along with a comparison to similarly sized corpora samples from the spoken and written domains; 2) Our part-of-speech tagger performance on the chat domain based on both the training data we used as well as the algorithms we employed; and 3) Chat dialog act classification results based on both the features we selected to measure as well as the algorithms we employed. Finally, in Chapter V we provide a summary of our work along with recommendations for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

In this chapter we review a broad body of work related to chat and Natural Language Processing (NLP) techniques. First, we will examine chat from a linguistic perspective, and highlight its similarities and differences to written and spoken language domains. Then, we will cover how chat is used today, and identify how NLP can be used to address requirements of applications that support its legitimate use (or detect its illegitimate use). Finally, we will provide a brief history of NLP techniques that we will apply to our chat research, to include part-of-speech tagging and dialog act classification.

A. LINGUISTIC STUDY OF CHAT

Before we start our discussion on the linguistic study of chat, we must first provide a common frame of reference with regards to chat itself. Although several chat protocols and applications abound, all contain variants of the following three features. First, there is a frame that displays all current participants in the particular session. This frame is updated as participants log on/off to the chat "room", and is publicly viewable to all currently in the room. Second, there is a frame displaying all posts submitted by all chat participants in the order that they arrived at the server. Thus, this main dialog frame is a record of all the (often interlaced) conversation threads that have taken place since the individual participant logged on to the room, and as such is also publicly viewable. Finally, there is a frame that is used for editing each participant's posts to the main dialog frame. Unlike the other two frames, though, this editing area is not publicly viewable. Only once the individual hits "Enter" do the contents of the editing frame become visible to the other participants in the main dialog frame.

Note that these limits on how chat is technically implemented are what give chat its near-synchronous quality. Often, one participant will respond to an earlier post at nearly the same time as another participant (to include the original poster) is responding. However, the application can only post those responses in the main dialog frame one at a time. This results in an interlacing effect among posts, even within a single conversation thread.

With these observations in mind, we are now ready to provide a brief review of the study of chat from a linguistic perspective. We first introduce a theoretical approach to communication in the chat domain, to include how traditional language constructs are modified for use in a domain enabled as well as restricted by technology. We then present an empirical study that explicitly compares language features used in a specific context (political discussion) across the written, spoken, and chat domains.

1. Zitzen and Stein's Linguistic Theory for Chat

Zitzen and Stein present a linguistic theory for chat founded in its pragmatic, social, and discourse communication properties [5]. As such, a primary objective of their research was to ascertain whether chat is simply a combination of written and spoken language, or if its properties are unique enough such that it constitutes a new genre within the CMC domain. Their theory is based in part on part on observations taken from three different chat sessions, which comprised a total of seven hours and eight minutes of verbal interaction and 12,422 words (not including words from system-generated messages).

One of the key features that can be used to distinguish chat from written and spoken domains is Nystrand's notion of *Context of Production* and *Context of Use* [6]. In particular, how *Context of Production* and *Context of Use* relate to one another across space and time help differentiate between the domains. As Zitzen and Stein observe,

> In face-to-face [spoken] conversations where the participants are physically co-present, *Context of Production* and *Context of Use* are concurrent. In other words, co-present participants can monitor another person's speech [and other physical cues] as it develops. Traditional written discourse is characterized by the spatiotemporal separation of *Context of Production* and *Context of Use* [5].

For chat, the aforementioned private editing frame functions as *Context of Production*, while the public main dialog frame functions as *Context of Use*. Since typing and editing a message cannot be monitored by the other chat participants, *Context of Production* is divorced from *Context of Use*. Thus, from a *Context of Production* perspective, chat is closer to written language, since per Zitzen and Stein "…there is no

incremental top-down, anticipatory processing of the auditory sound material…[as well as] paralinguistic information [5]." That being said, *Context of Use* for chat is much closer to that of spoken language as compared to written discourse such as letter writing or even email.

Another concept that can be used to differentiate chat from the other domains is the conversational concept known as turn-taking. Turn-taking is the process that determines who gets to "hold the floor" in a conversation. Sacks et al proposes the following "algorithm" (presented in Table 1) that is used by spoken conversation participants to allocate turns. Further, Sacks et al asserts this algorithm generates two driving forces in spoken conversation: avoidance of silence and avoidance of overlapping talking [7].

1. The current floor holder may implicitly or explicitly select the next speaker, who is then obliged to speak. 2. If the current floor holder does not select the next speaker, the next speakership may be self-selected. The one who starts to talk first gets the floor. 3. If the current speaker does not select the next speaker, and no self-selected speakership takes place, the last speaker may continue. 4. If the last (current) speaker continues, rules 1-3 reapply. If the last (current) speaker does not continue, then the options recycle back to rule 2 until speaker change occurs.

Table 1. Turn Allocation Techniques in Spoken Language (From [7])

Zitzen and Stein assert that in chat conversations "a much more intricate and complicated layering of partial [turn-taking] mechanisms" replace those of Sack's et al turn allocation algorithm for spoken conversations [5]. First, the speaker-selection properties described in Table 1 are replaced with a "first message to server, first message posted to dialog frame" concept. Thus, technology (and not personal relations and face management) determines who obtains the floor in chat.

Second, Zitzen and Stein assert that the concept of being a "hearer" or "speaker" in chat is much more complex than that in spoken conversation [5]. They note that Garcia and Jacobs observed

> A [chat] participant can be a waiter and a reader at the same time, both waiting for a response to a previous post and simultaneously reading or scrolling through previous postings. A typing participant who is awaiting a response to an earlier message is both a waiter and message constructor [8].

Again, chat technology permits participants to play multiple roles at the same time.

Regarding silence, Zitzen and Stein note that lapses in conversation are socially stigmatized in spoken conversation, with one of the effects being to cause participants to engage in small talk just to keep conversation going. In chat, silence can be characterized by two types: total silence, where there are no postings at all; and selective silence, where a participant does not respond to a post addressed to him/her [5]. Zitzen and Stein assert that, as in spoken conversation, silence in chat, although also not desirable, is not as socially damaging [5]. Again, technology plays a role in the greater acceptance of (or forgiveness for) silence in chat. Instead of responding to a post directed to him/her, Zitzen and Stein state that a chat participant may be "reading [other] incoming messages, scrolling through previous logfiles, waiting for a response, and even typing a message [5]." That being said, they note that chat participants do make active attempts to forewarn others of activities that may be misconstrued as silence.

> Entering a chat is less obliging than entering a conversation in the sense that the participants in a conversation have to stay until there is some negotiated and agreed upon closing procedure. Contrary to face-to-face situations where participants are rather hesitant to leave the room in the middle of an ongoing conversation, in chats we find constant coming and going, frequently accompanied by the use of the acronym BRB (be right back) [also AFK, "away from keyboard"] which functions as a meta-communicative attempt. [5]

Lurking, a feature that appears to be unique to chat and other forms of CMC, is the concept of silence taken to the extreme, with the chat participant never contributing to the ongoing dialog. Zitzen and Stein note that in spoken conversation, there is a strict boundary between those that participate and those that do not [5]. Although eavesdropping certainly occurs in conversations, the eavesdropper is not a ratified conversation participant. In spoken conversation, participants must go through a process of acceptance, where newcomers must first negotiate their entry. In chat, technology

handles this negotiation process, with system messages indicating "User X has entered/left the room" and the application's participant frame indicating to all who is currently in the room. That being said, Zitzen and Stein note that as the number of participants increase, so does the potential for successful lurking, since the presence of the lurker is forgotten due to their lack of dialog contribution as well as their disappearance in the "sea" of participants in the application's participant frame [5]. Once lurking has been detected in chat, it is usually confronted and criticized. Indeed, chat applications now have the ability for users with administrator-like privileges to "kick" lurkers out of the room.

With these considerations in place, Zitzen and Stein define two states for chat conversational presence: lurking (second order presence), and composing/appearing on screen (first order presence) [5]. In other words, "not messaging a word means being virtually absent, while more frequently messaging establishes a perception of presence [5]." However, as in spoken conversation, there is an expected level of contribution among participants. As mentioned earlier, silence is undesirable, yet too much contribution is regarded as "hogging the conversation". Thus, first order chat participants feel the need to regulate both the number of posts they make as well as their length. Zitzen and Stein elaborate

> Longer messages do not only take a longer to type, but they also occupy more space in the public dialog box, at the same time pushing away other participant's contributions, which in turn decreases the other one's virtual presence…Shorter messages are not only less time-consuming with regard to production, waiting, and reception, they also help to place a message as adjacent as possible to a previous message, a participant wishes or is asked to respond to [5].

Thus, chat participants must balance their level of verbal activity to maintain mutual presence within the ongoing conversation. Given chat's technical considerations, participants achieve this in part through what Zitzen and Stein have defined as the "split turn," where a single contribution "utterance" is broken up into two or more posts [5]. Based on their data, Zitzen and Stein categorize the split turn phenomenon into four different types, with their construction employing different linguistic techniques. These

techniques include (but are not limited to) continuation posts starting with transitional relevance place words such as conjunctions and prepositions; the use of ellipses (…) at the end of messages to indicate more to come; multiple successive posts by the same participant, each addressing a different topic and/or participant; etc.

With this description of Zitzen and Stein's theory of chat complete, we need to address how their observations potentially impact an NLP application. We believe that split turns have a definite impact on how NLP applications handle chat text at the lexical, syntactic, and semantic levels, particularly if the application intends to use data from non-chat domains to train on. From as lexical perspective, a word's part-of speech tag is dependent in part on its context; words near the boundaries of split turns lose part of that context. Similarly, potential syntactic productions (e.g. noun phrases expanding to nouns and prepositional phrases) are lost when those productions occur across split turn boundaries. Finally, the full meaning of a single utterance requires access to all split turns that it comprises. Thus, an NLP application for the chat domain must have a way to both identify split turns and, as necessary, combine those that represent a single utterance.

With our discussion of Zitzen and Stein complete, we now turn Freiermuth's explicit comparison between chat and counterparts within the written and spoken domains.

2. Freiermuth's Comparative Analysis of Chat, Written, and Spoken Texts

In his Ph.D. dissertation, Freiermuth explicitly compared chat with traditional written and spoken language from the same content domain—political discussion [9]. To maintain consistency, he selected 3000 words for each type of communication. For the spoken domain, he used the first 500 transcribed words (excluding the monologue) from six different episodes of *Politically Incorrect*, a late-night television program. For the written domain, he used samples from the editorial section of the *Standard-Times*, a newspaper which serves the south coast of Massachusetts. Finally, for chat Freiermuth collected samples from one of the political chat channels on America Online, entitled *From the Left*.

Freiermuth used grammatical and functional features identified by Chafe and Danielewicz's cognitive approach to compare the three domains [10]. These features can be grouped into five categories: 1) Vocabulary variety; 2) Vocabulary register; 3) Syntactic integration; 4) Sentence-level conjoining; and 5) Involvement and detachment. A description of these categories, the specific features used, and a summary of Freiermuth's findings for how chat compares with spoken and written domains follows.

Vocabulary variety refers to the size of the vocabulary used in the particular domain [9]. Under this category, Freiermuth measured type/token ratios, or the total number of words in the sample divided by the number of unique words in the sample; hedging, reflecting when the participant is dissatisfied with the lexical choice ("sort of" and "kind of"); and inexplicit third person use ("it", "this", and "that") that have no clearly identified antecedent. Based on these measurements, Freiermuth had the following conclusions.

> [First,] Chatters have more time to choose appropriate vocabulary when compared to speakers. [Second,] Chatters increase variety by using creative and innovative language forms, as well as addressivity. [Third,] Chatters do not use hedges, indicating they are either satisfied with their language choices or that they do not care if they are imprecise because they cannot be held accountable for what they say [9].

Vocabulary register, or level, refers to the types of words that are common in spoken versus written settings [9]. Under this category, Freiermuth specifically measured literary language use, or the number of words that are not considered usual in typical spoken language (e.g. "elaborate" and "introspection"); colloquial language use, or the number of words that appear lexically fresh, i.e. change over time (e.g. "chill out"); and contractions. Based on these measurements, Freiermuth had the following conclusions.

> [First,] Chatters have less time than writers (much), but more time than speakers. Their cognitive processing of language is not under the same heavy demands that speakers face. [Second,] Chatters tend to mimic spoken language, but because they are aided by time, they sometimes elevate their language sophistication [9].

Syntactic integration refers to a strategy employed primarily by writers to incorporate linguistic elements into clauses to be more concise and precise while

expanding intonation [9]. Under this category, Freiermuth specifically measured prepositions and stringed prepositions; complex causal conjoining; locative and temporal adverbs; and preposed attributive adjectives and noun modifiers. Based on these measurements, Freiermuth had the following comments and conclusions.

> [First, depending on the chat application,] Chatters are limited by their environment. AOL restricts the number of characters a participant may type per turn, so integration is not a useful strategy. [Second,] Chatters must cope with many simultaneous difficulties, while trying to be an active member of the conversation. The complex dynamics of Internet chat (e.g., the number of chatters, the problem of intervening turns from multiple conversations, the difficulties of processing text embedded in the midst of dialogic interaction, etc) do not warrant expanding units. [Third,] Chatters are capable of more complex clausal interaction, but prefer speed to precision [9].

Sentence level conjoining refers to using conjunctions to join smaller sentences into larger ones. Freiermuth states that speakers primarily use this as a way to both establish and maintain the floor in a conversation as well as to organize their thoughts [9]. For this category, Freiermuth's data indicated that chat text was more like written text based on the following rationale.

> [First] Chatters have no need to establish or maintain the floor because they construct dialog simultaneously with other chatters who are online. In other words, the floor is always available to them. [Second,] Chatters do not need to organize their thoughts within the framework of a conversation. They can take as much time as they want without affecting conversational dynamics [9].

The final category refers to the observation that written language is usually more detached, while spoken language is usually more involved. Under this category, Freiermuth specifically measured the number of "you/ya knows"; the number of first, second, and third person pronouns; indicators of probability, such as "normally" and "possibly", which permit the communicate an escape from culpability; and the use of passives and addressivity, which refer to degree with which the communicator indicates a concrete "doer" for a particular action [9]. Based on these measurements, Freiermuth had the following conclusions.

> [First,] Chatters have no need to cue interlocutors with classic discourse markers. In fact, such markers would probably have little effect on the participants online. [Second,] Chatters tend not to respond to questions. They cannot be held accountable if they fail to answer questions, and it is likely the problem of intervening turns causes them to forget to answer questions. [Third,] Chatters use second person pronouns at about the same frequency as [spoken] conversationalists, but they tend to use them in a more confrontational way, while conversationalists use them in a generic sense quite frequently. [Fourth,] Chatters must use addressivity to target a particular chatter that is online; otherwise, it is quite difficult to identify who is chatting to whom [9].

With Freiermuth's observations in mind, we need to address how they potentially impact an NLP application tailored for use with chat. Obviously, chat has features of both spoken and written language. For example, chatters exhibit the vocabulary diversity of written communicators. And yet, Freiermuth notes that they prefer not to expand clausal units the way written authors do, instead favoring speed over precision [9]. As such, if chat-specific training data is limited for an NLP application, it would seem to make sense to make use of training data from both spoken and written domains. An interesting question would be if there is a preferred ratio of spoken to written training data that optimally mimics chat. Furthermore, depending on the NLP application, one type of data might be preferred over the other. Using our examples above, since chat is closer to written language in terms of vocabulary size, training data from the written domain might be preferred for a part-of-speech tagging application. However, since posts are less complex structurally in chat compared to the written domain (as evidenced by lack of clausal unit expansion), then perhaps transcribed spoken text might be better for syntax parsing.

With our brief overview of the linguistic study of chat complete, we now turn to how chat is being used (and misused) today.

B. CHAT USE TODAY

In this section we introduce two uses of NLP in chat today: 1) Military use in support of tactical command and control (C2) processes; and 2) Detecting illegitimate

chat use. In both cases we provide examples of high level chat application requirements, and identify how NLP can be used to meet those requirements.

1. Tactical Military Chat

In his master's degree thesis, Eovito explored the impact of synchronous, text based chat to fill gaps in military systems for tactical C2 [2]. Eovito notes that, as is the case with many military systems, the use of chat for tactical C2 evolved in an ad hoc fashion. As such, there has never been a formal requirements analysis of text-based chat tools either from a top-down ("What C2 deficiencies are addressed by chat tools?") or bottom-up ("What capabilities do chat tools bring to the war fighter?") perspective. A primary objective of Eovito's research was to develop such a set of requirements to help guide the development of next-generation C2 systems.

To develop requirements for military tactical chat, Eovito first administered both surveys and interviews to establish a set of use cases. Eovito solicited responses from users spanning all four U.S. military services as well as Canadian, Australian, and New Zealand coalition forces. The settings where those users employed tactical chat spanned major combat such as Operations Enduring Freedom (OEF) and Iraqi Freedom (OIF) to military operations other than war (MOOTW) such as Hurricane Katrina relief.

From these use cases, Eovito then extracted a framework for tactical chat requirements. The framework consisted of four categories: Functionality, Information Assurance, Scalability, and Interoperability. A complete list as well as description of tactical chat requirements in all categories can be found in [2].

NLP techniques are critical to fully address many of the functional requirements depicted in Table 2. For example, thread population/repopulation, a core requirement, consists of the ability for users to select a portion of the chat log (i.e., conversation thread) to repopulate in the event of late entry into a chat session. Such an automated feature requires the system to select only the subset of posts within the overall session that comprise the specific thread. Semantic and discourse NLP techniques are vital in accomplishing this task. Similarly, foreign language text translation requires NLP techniques that can identify idioms across languages (e.g. "bogey moving like a bat out of hell!") and translate accordingly.

1. Participate in Multiple Concurrent Chat Sessions*
2. Display Each Chat Session as Separate Window
3. Persistent Rooms & Transitory Rooms*
4. Room Access Configurable by Users
5. Automatic Reconnect & Rejoin Rooms*
6. Thread Population/Repopulation*
7. Private Chat "Whisper"*
8. One-to-One IM (P2P)
9. Off-line Messaging
10. User Configured System Alerts
11. Suppress System Event Messages
12. Text Copying*
13. Text Entering*
14. Text Display*
15. Text Retention in Workspace*
16. Hyperlinks
17. Foreign Language Text Translation
18. File Transfer
19. Portal Capable
20. Web Client
21. Presence Awareness/Active Directory*
22. Naming Conventions Identify Functional Position*
23. Multiple Naming Conventions
24. Multiple User Types
25. Distribution Group Mgmt System for Users
26. Date/Time Stamp*
27. Chat Logging*
28. User Access to Chat Logs*
29. Interrupt Sessions
(* denotes a core requirement)

Table 2. Consolidated Functional Requirements for Tactical Military Chat (From [2])

Similarly, NLP can play a role in meeting information assurance requirements as depicted in Table 3. For example, Eovito notes that many user IDs in the various sessions are functional, making it difficult to know who is really in the chat room.

However, NLP can be used to identify characteristics of an author's language use, thus supporting user authentication. In addition, NLP lexical and semantic techniques can assist in permitting authorized transfer of information across domains within a security level (e.g. Joint vs. Coalition information) as well as across levels (e.g., Secret vs. Top Secret).

1.	Login and User Authentication
2.	Access Control
3.	User Authentication by Active Directory
4.	Unique ID for all users worldwide
5.	PKI Enabled (DOD Common Access Card)
6.	Provide Encryption
7.	Network Security Tools
8.	Cross Security Domain Functionality
9.	Multi-Level Security Operation
10.	Cross Security Domain Functionality

Table 3. Consolidated Information Assurance Requirements for Tactical Military Chat (From [2])

Eovito concludes with recommendations for follow on research in the following categories: 1) Chat data mining; 2) Net-Centric Enterprise Services; 3) Extensible Markup Language (XML); 4) Human Factors; 5) Specific War Fighting Doctrine; and 6) Information Assurance [2]. We have already discussed how NLP plays a role with information assurance. That being said, NLP techniques can improve the performance of data mining, where semantic and discourse clues can help narrow the search space for a particular thread. Similarly, many human factor concerns must be addressed by NLP, which can improve the human system interface by permitting humans to "command" the chat system with natural language.

We have demonstrated how NLP techniques can play a role in improving the legitimate use of chat in a military context. We now examine how they can be used by law enforcement and intelligence analysts to detect illegitimate use of chat.

2. Detecting Illegitimate Chat Use

In her master's thesis, Lin provides motivation for the study of chat and associated behavior [11]. As with any new technology, there is potential both for the

betterment of and detriment to society, and the Internet is no exception. Not only does Internet-based chat permit people to communicate for both business and pleasure, it is also a medium with great potential for misuse. Lin specifically notes how Internet-based chat has exacerbated the problem of sex crimes committed against children. In addition, she postulates how chat can be used by terrorists to communicate, thus enhancing planning, command, and control for terrorist groups as well as other criminal organizations.

In response to this, Lin proposed that authorship attribution techniques can be used to automatically detect whether chat is being abused in a particular setting [11]. To put her theory to test, she collected 475,000+ posts made by 3200+ users from five different age-oriented chat rooms at an Internet chat site. The chat rooms were not limited to a specific topic, i.e. were open to discussion of any topic. Lin's goal was to automatically determine the age and gender of the poster based on their chat "style" as defined by features of their posts. Thus, if a particular user in a teen-oriented chat room made posts with features associated with an adult male, this information could be used by authorities to more closely scrutinize this user's behavior.

The specific features Lin captured for each post were surface details, namely, average number of words per post, size of vocabulary, use of emoticons, and punctuation usage [11]. Lin relied on the user's profile information to establish the "truth" of each user's age and gender. Lin then used the Naïve Bayes machine-learning method (described in greater detail in Chapter III) to automatically classify the user's age and gender based on the aforementioned features of all the posts the user made.

Lin's work represents a significant, albeit initial, effort to apply NLP techniques specifically to chat to determine author characteristics. Although her results were mixed, better surface features (e.g. distribution of all words used instead of just emoticons and punctuation) as well as "hidden" features (e.g. syntactic structure of the posts) have the potential to improve authorship classification accuracy.

With our brief description of how NLP can be used in chat applications, we now turn to the linguistic study of the chat domain.

C. NATURAL LANGUAGE PROCESSING TECHNIQUES

In this section we provide a brief review of the natural language processing techniques we will use in our research on chat. We first introduce both the concept and specific examples of corpora labeled with meta-information. We then discuss automated part-of-speech tagging, to include specific techniques that have been developed, how it supports higher level NLP applications, and factors that influence its performance. Finally, we present automated dialog act classification, to include its use in NLP applications, exhaustive results from a spoken domain, and initial results in the CMC domain. Note that in this section we limit our targeted NLP review to a historical perspective. We discuss the specific technical implementation of automated part-of-speech tagging and dialog act classification methods in Chapter III.

1. Annotated Corpora

State-of-the-art natural language processing applications rely on labeled data for training. Over the years, numerous corpora annotated with lexical, syntactic, and semantic "meta-information" have been developed for such purposes. One of the first corpora available to the larger NLP research community was developed in the 1960s by Francis and Kucera at Brown University [12]. Commonly referred to today as the Brown Corpus, it contained over one million words collected from 500 samples written by native speakers of American English and first published in 1961. The samples from the 15 genres are shown in Table 4.

1.	*Press: Reportage* (44 texts: Political, Sports, Society, Spot News, Financial, Cultural)
2.	*Press: Editorial* (27 texts: Institutional Daily, Personal, Letters to the Editor)
3.	*Press: Reviews* (17 texts: Theatre, Books, Music, Dance)
4.	*Religion* (17 texts: Books, Periodicals, Tracts)
5.	*Skill and Hobbies* (36 texts: Books, Periodicals)
6.	*Popular Lore* (48 texts: Books, Periodicals)
7.	*Belles-Lettres: Biography, Memoirs, etc* (75 texts: Books, Periodicals)
8.	*Miscellaneous: US Government & House Organs* (30 texts: Government Documents, Foundation Reports, Industry Reports , College Catalog, Industry House organ)
9.	*Learned* (80 texts: Natural Sciences, Medicine, Mathematics, Social and Behavioral Sciences, Political Science, Law, Education, Humanities, Technology and Engineering)
10.	*Fiction: General* (29 texts: Novels, Short Stories)
11.	*Fiction: Mystery and Detective Fiction* (24 texts: Novels, Short Stories)
12.	*Fiction: Science* (6 texts: Novels, Short Stories)
13.	*Fiction: Adventure and Western* (29 texts: Novels, Short Stories)
14.	*Fiction: Romance and Love Story* (29 texts: Novels, Short Stories)
15.	*Humor* (9 texts: Novels, Essays, etc.)

Table 4. Brown Corpus Description (From [12])

The original corpus contained only the words themselves. Later, 87 part-of-speech tags were applied to the corpus, permitting a variety of statistical analysis on the texts themselves as well as providing training data for NLP applications. Because of its widespread availability to researchers, the Brown corpus became a de facto standard model for the English language.

Seeking to institutionalize the availability of corpora such as Brown, the Linguistic Data Consortium (LDC), first mentioned in Chapter I, was founded with a grant from the Defense Advanced Research Projects Agency [4]. Such corpora are expensive to create, maintain, and distribute; thus, the service provided by LDC enables replication of published results, supports a fair comparison of algorithms, and permits individual users to make corpora additions and corrections. Since many of the data contributions are copyrighted, the LDC distributes them for the purposes of research, development, and education through more than 50 separate Intellectual Property Rights (IPR) contracts.

It is interesting to note that LDC comments on the future requirements for linguistic technology. Specifically,

> We humans spend much of our lives speaking and listening, reading and writing. Computers, which are more and more central to our society, are already mediating an increasing proportion of our spoken and written communication—in the telephone switching and transmission system, in electronic mail, in word processing and electronic publishing, in full-text information retrieval and computer bulletin boards, and so on [4].

However, as noted in Chapter I, of the 381 corpora provided by the LDC, only three contain samples from the computer-mediated communication domain: LDC2006T06 (ACE 2005 Multilingual Training Corpus, which contains newsgroup and weblog samples); LDC2006T13 (Google's Web 1T 5-gram Version 1); and LDC2007T22 (2001 Topic Annotated Enron Email Data Set) [4]. If we seek to build NLP applications that support CMC such as chat, we require a certain amount of data from the domain itself.

With our brief discussion on the role corpora play in state-of-the-art NLP applications in general, we now turn to an important component of such applications: part-of-speech tagging.

2. Part-of-Speech Tagging

Part-of-speech tagging is the process of assigning a part-of-speech label (e.g. verb, noun, preposition, etc) to a word in context based on its usage. Several higher order NLP applications rely on part-of-speech tagging as a preprocessing step. For example, both [13] and [14] note that information retrieval applications make use of part-of-speech tagging, which often involves looking for nouns and other important words that can be identified in part by their part-of-speech tag. Indeed, the dialog act classification application that we developed for chat incorporated some features based on word part-of-speech tags. As such, part-of-speech tagging is an important topic to discuss when applying NLP techniques to a heretofore unexplored domain.

a. Algorithmic Approaches

Jurafsky and Martin describe three classes of algorithms for part-of-speech tagging [14]. The first class, commonly referred to as rule-based taggers, rely on a two phase approach assign tags. In the first phase, a dictionary is used to assign each word a set of potential parts-of-speech. The second phase uses large lists of hand-written rules that are successively applied to reduce the set to a single tag. One rule-based tagging approach, referred to as the English Constraint Grammar (EngCG), reports accuracies of 99%, although not all ambiguities are resolved, i.e. EngCG sometimes returns a set that includes more than one tag.

The second class, referred to as stochastic taggers, use probabilities based on counts of words and their tags from a training corpus [14]. Stochastic taggers include n-gram-based tagging approaches as well as Hidden Markov Models (HMMs), which differ based on the varying degrees of context considered by the algorithms. We present a full description of the technical details for both stochastic tagging approaches in Chapter III. HMM-based tagging approaches report accuracies of 95-96%.

The final class, known as Brill Transformational-Based Learning tagging, is essentially a combination of the previous two classes [14]. As with rule-based tagging, the algorithm uses rules successively applied to initially assign and later refine part-of-speech tags. However, like stochastic taggers, the rules are learned based on the frequency of their successful application within a training corpus. We present a full description of the Brill rule templates and learning algorithm in Chapter III. Brill reports tagging accuracies of 96-97% using this approach [15].

With our discussion of tagging approaches complete, we now look at how they work in conjunction with annotated corpora to affect overall tagging performance.

b. Performance Factors

Manning and Schütze note that the performance of part-of-speech taggers is greatly influenced by four factors [13]. We note these factors, along with their potential effect on a part-of-speech tagger crafted specifically for the chat domain. The

first, the amount of training data available, is straightforward: the more data available to train on, the better the accuracy of the tagger. Since no publicly available tagged corpus currently exists for the chat domain, one will have to be created. By its very nature, then, this will be a resource-intensive activity, and as such will initially be much smaller than those available for the more established written and spoken domains.

The second factor is the tag set used [13]. Although a larger tag set permits a more fine-grained determination for a particular word in context, this very fact leads to the potential for more ambiguity of the given word. Thus, if a corpus is tagged with two tag sets, as is the case with the Brown corpus (original Brown 87 POS tag set and later Penn Treebank 45 POS tag set), taggers using the same algorithm will generally have a higher accuracy on the corpus tagged with the smaller tag set. Therefore, when tagging a chat domain corpus, we would prefer to use a smaller, established tag set. That being said, the chat domain contains features such as emoticons (e.g., ":-)", a smiley face on its side) that do not exist in other domains. As such, we would need to decide if an existing tag appropriately describes emoticon usage, or if instead a new tag should be created.

The third factor is the difference between the training corpus and the corpus of application [13]. If the training and application text are drawn from the same source, accuracy will be high. This is generally the case for the highest accuracy taggers described in the literature. However, as alluded to in the LDC quote from Chapter I, if training and application text are from a different domain, accuracy can be poor. Thus, the task of building a highly accurate POS tagger for the chat domain is complicated by the fact that currently tagged corpora are from significantly different domains. Experiments that consider tagging accuracy on chat based on the training domain are presented in Chapter IV.

The final factor affecting tagging accuracy is the occurrence of unknown words [13]. Obviously, the more words encountered in application that have not been seen during training, the more tagging performance will suffer. This may play a particularly important role in the chat domain, where misspellings as well as the use of emphasis through character repetition (e.g., "riiiiiiggggghht" for "right") are frequently encountered.

With our overview of part-of-speech tagging tools complete, we now turn our discussion to a higher-order NLP task—dialog act modeling.

3. Dialog Act Modeling

The dialog act, per Austin, represents the meaning of an utterance at the level of illocutionary force [16]. In layman's terms, dialog act classification categorizes a conversational element into classes such as "statements", "opinions", "questions", etc. Thus, dialog acts provide a first level of analysis for discourse structure.

Dialog act modeling has a wide number of potential applications. As described by Stolcke et al, a meeting summarization application needs to keep track of who said what [17]. Similarly, a telephone-based conversational agent needs to know if it was asked a question or tasked to do something. Indeed, Stolcke et al demonstrated that dialog act labels could be used in a speech recognition system to improve word recognition accuracy by constraining potential recognition hypotheses. Dialog acts might also be used to infer the types of relationships (e.g. superior to subordinate versus peer to peer) that occur within a social network. Finally, as applied to chat, dialog acts could be used to help separate interleaved conversation threads.

With this definition of dialog acts and their potential applications introduced, we now turn to a brief overview of Stolcke et al's in-depth research concerning dialog act modeling in conversational speech and its subsequent adaptation to two computer-mediated communication genres.

a. Spoken Conversation

Stolcke et al used dialog acts to model the utterances within conversational speech [17]. The conversational speech domain was represented by 1,155 conversations (to include both sound waveforms and transcribed text) drawn from the Switchboard corpus of spontaneous human-to-human telephone speech. The 42 dialog acts along with an example and its frequency of occurrence within Switchboard are presented in Table 5.

Tag	Example	Percent
Statement	Me, I'm in the legal department.	36%
Backchannel/Acknowledge	Uh-huh.	19%
Opinion	I think it's great.	13%
Abandoned/Uninterpretable	So, -/	6%
Agreement/Accept	That's exactly it.	5%
Appreciation	I can imagine.	2%
Yes-No-Question	Do you have to have any special training?	2%
Non-Verbal	<Laughter>,<Throat_clearing>	2%
Yes Answers	Yes.	1%
Conventional-Closing	Well, it's been nice talking to you.	1%
Wh-Question	What did you wear to work today?	1%
No Answers	No.	1%
Response Acknowledgement	Oh, okay.	1%
Hedge	I don't know if I am making any sense or not.	1%
Declarative Yes-No-Question	So you can afford to get a house?	1%
Other	Well give me a break, you know.	1%
Backchannel-Question	Is that right?	1%
Quotation	You can't be pregnant and have cats.	0.5%
Summarize/Reformulate	Oh, you mean you switched schools for the kids.	0.5%
Affirmative Non-Yes Answers	It is.	0.4%
Action-Directive	Why don't you go first	0.4%
Collaborative Completion	Who aren't contributing.	0.4%
Repeat-Phrase	Oh, fajitas	0.3%
Open-Question	How about you?	0.3%
Rhetorical-Questions	Who would steal a newspaper?	0.2%
Hold Before Answer/Agreement	I'm drawing a blank.	0.3%
Reject	Well, no	0.2%
Negative Non-No Answers	Uh, not a whole lot.	0.1%
Signal-Non-Understanding	Excuse me?	0.1%
Other Answers	I don't know.	0.1%
Conventional-Opening	How are you?	0.1%
Or-Clause	or is it more of a company?	0.1%
Dispreferred Answers	Well, not so much that.	0.1%
3rd-Party-Talk	My goodness, Diane, get down from there.	0.1%
Offers, Options, & Commits	I'll have to check that out	0.1%
Self-Talk	What's the word I am looking for	0.1%
Downplayer	That's all right.	0.1%
Maybe/Accept-Part	Something like that	< 0.1%
Tag-Question	Right?	< 0.1%
Declarative Wh-Question	You are what kind of buff?	< 0.1%
Apology	I'm sorry.	< 0.1%
Thanking	Hey, thanks a lot	< 0.1%

Table 5. 42 Dialog Act Labels for Conversational Speech (From [17])

Stolcke et al's model detects and predicts dialog acts based on lexical, collocational, and prosodic (e.g. sound waveform pitch, duration, energy, etc) features of the utterance as well as the overall discourse coherence of the sequence itself [17]. This overall discourse structure was treated as a Hidden Markov Model, with the specific utterances representing the observation sequence "emitted" from the dialog act state sequence. Constraints for the likely dialog act sequence were modeled with dialog act n-grams, which were combined with n-grams, decision trees, and neural networks modeling lexical and prosodic features of the dialog act itself. Stolcke et al achieved accuracy of results of 65% (based on automatic speech recognition of words combined with prosody clues) and 71% (based on word transcripts), compared to a chance baseline accuracy of 35% and human accuracy of 84% [17].

b. Computer-Mediated Communication

Drawing from Stolcke et al, Wu et al used the dialog act methodology to model the postings in online chat conversations [18]. For their research, the chat domain was represented by nine different Internet Relay Chat (IRC) conversations containing a total of 3,129 posts. The 15 dialog acts along with an example and its frequency of occurrence within the data set are presented in Table 6.

Tag	Example	Percent
Statement	I'll check after class	42.5%
Accept	I agree	10.0%
System	Tom [JADV@11.22.33.44] has left#sacbal	9.8%
Yes-No-Question	Are you still there?	8.0%
Other	**********	6.7%
Wh-Question	Where are you?	5.6%
Greet	Hi, Tom	5.1%
Bye	See you later	3.6%
Emotion	lol	3.3%
Yes-Answer	Yes, I am.	1.7%
Emphasis	I do believe he is right.	1.5%
No Answer	No, I'm not.	0.9%
Reject	I don't think so	0.6%
Continuer	And …	0.4%
Clarify	Wrong spelling	0.3%

Table 6. 15 Post Act Classifications for Chat (From [18])

Wu et al's post act classifications were based on a set of rule templates learned via Brill's Transformational Based Learning algorithm [18]. Based on nine-fold cross validation of all posts, Wu achieved an average accuracy of 77.56% (maximum = 80.89%, minimum = 71.20%). In Chapter III, we discuss how we used the Wu et al tag set (with minor interpretation differences) to perform chat dialog act modeling on our data set of chat posts.

Ivanovic also drew heavily from Stolcke et al's work to assign dialog acts to instant messaging (IM) sessions [3]. Unlike Stolcke and Wu's domains, which were conversational in nature, Ivanovic's domain was task-oriented dialog represented by online shopping assistance provided by the MSN Shopping web site. Specifically, the data set consisted of nine chat sessions, totaling 550 utterances and 6,500 words. The 12 IM dialog acts along with an example and its frequency of occurrence within the data set are presented in Table 7.

Tag	Example	Percent
Statement	I am sending you the page now	36.0%
Thanking	Thank you for contacting us	14.7%
Yes-No-Question	Did you receive the page?	13.9%
Response-Ack	Sure	7.2%
Request	Please let me know how I can assist	5.9%
Open-Question	how do I use the international version?	5.3%
Yes-Answer	yes, yeah	5.1%
Conventional-Closing	Bye Bye	2.9%
No-Answer	no, nope	2.5%
Conventional-Opening	Hello Customer	2.3%
Expressive	haha, :-), grr	2.3%
Downplayer	my pleasure	1.9%

Table 7. 12 Dialog Act Labels for Task-Oriented Instant Messaging (From [3])

In contrast to Wu et al, who applied a single dialog act to each post, Ivanovic segmented dialog acts at the utterance level [3]. As such, utterances and their associated dialog act can either span multiple posts or reside next to zero or more utterances within a single post. After utterance segmentation, Ivanovic resynchronized the utterances since IM (like chat) exhibits a certain amount of asynchronicity due to the technology associated with posting. Ivanovic's machine-learning model combined the Naïve Bayes classifier with n-grams (n= 1, 2 and 3). Based on nine-fold cross validation of all utterances, Ivanovic achieved an average bigram (n = 2) model accuracy of 81.6% (maximum = 92.4%, minimum = 75.0%) [3].

With our review of chat and associated NLP applications complete, we now turn to the technical details associated with our research.

III. TECHNICAL APPROACH

In this chapter we will cover the technical approach we used in building the corpus as well as the technical details associated with both part-of-speech tagging and chat dialog act classification methodologies.

A. BUILDING THE CORPUS

In this section we will cover the details associated with building the corpus, to include its conversion to an Extensible Markup Language (XML) format; subsequent masking of participant names for privacy considerations; part-of-speech and chat dialog act labeling decisions; and the general bootstrapping process.

1. Data Conversion to XML

As mentioned earlier, Lin collected open topic chat dialog samples from five different age-oriented chat rooms [11]. These samples, taken over the course of 26 sessions in the fall of 2006, included session log on information as well as 477,835 posts made by the users as well as automated posts made by both the chat room system as well as "chatbots". Chatbots are automated user software independent of the chat room system that assist human participants, provide entertainment to the chat room, etc.

In addition to the sessions, Lin collected the chat room profiles on each of the approximately 3,200 users participating in the session dialog samples. The profiles often (but not always) contained a variety of information on the individual user, including age, gender, occupation, and location. The profile files were provided to us in an HTML format.

In order to enhance accessibility to this information for future researchers, we converted both the sessions as well as the profiles to an XML format using the Python ElementTree module [19]. In particular, we created two versions of the corpus. The first version included the entirety of the 26 sessions as well as a single file containing all users' age, gender, occupation, and location information, as available. In both the sessions and the profile file, the users were referred to by the original screen names collected by Lin.

2. Privacy Masking

In the second version of the corpus, we took a contiguous sample of approximately 700 posts from 15 of the 26 sessions each. In this version, however, we altered the users' names in each session such that they were referred to by a standard mask with a key representing the order they joined the session. For example, "killerBlonde51" would become "10-19-40sUser112" in the session collected from the 40s-oriented chat room on October 19; "11-08-40sUser23" in the session collected on November 8; and so on. Similarly, we sanitized the profile file with a single mask as well as a pointer to a list of masks that the particular user was referred to in the various session files. Using the previous example, killerBlonde51 would be referred to as "user57" in the profile file, referencing a list containing 10-19-40sUser112, 11-08-40sUser23, and any other session masks that killerBlonde51 participated in. To date, we have privacy-masked 10,567 of the 477,835 posts in this manner.

Why did we decide to perform privacy masking? If we are to make the corpus available to the larger research community, this must be accomplished. It was straightforward to replace the user's screen name in both the session samples as well as the profile file. However, more often than not, users were referred to by variations of their screen names in other users' posts. For example, other users would refer to killerBlonde51 as "killer," "Blondie," "kb51," etc. Although regular expressions can assist in the masking task, ultimately 100% masking required us to hand-verify that the appropriate masks had been applied in every post.

We should note that although masking is essential to ensure privacy, it results in a loss of information. For example, the way to which users are referred often conveys additional information, for example, familiarity and emotion; this information is lost in the masking process. In addition, we observed that a user's screen name would become a topic of conversation independent from the original user; again, the origin of this conversation thread is lost in the masking process.

Once we complete the masking process, we then turned to tokenizing the posts of the privacy-masked version of the corpus and annotating the tokens with part-of-speech tags.

3. Part-of-Speech (POS) Tagging

As discussed in Chapter II, several POS-tagged corpora in many languages are available to NLP researchers. The corpora we used to help train various versions of the taggers are contained within the Linguistic Data Consortium's Penn Treebank distribution [20]. The first corpus, referred to as Wall Street Journal (WSJ), contains over one million POS-tagged words collected in 1989 from the Dow Jones News Service. The second, briefly introduced in Chapter II and referred to as Switchboard, was originally collected in 1990 and contains 2,430 transcribed, POS-tagged, two-sided telephone conversations among 543 speakers from all areas of the United States. Each conversation averaged about six minutes in length, totaling 240 hours of speech and about three million words total. The third, also discussed in Chapter II and referred to from here on as Brown, consists of over one million POS-tagged words collected from 15 genres of written text originally published in 1961.

BES	"'s" contraction for "is" *	SYM	Symbol
CC	Coordinating conjunction	TO	"to"
CD	Cardinal number	UH	Interjection
DT	Determiner	VB	Verb, base form
EX	Existential there	VBD	Verb, past tense
FW	Foreign word	VBG	Verb, gerund or present participle
HVS	"'s" contraction for "has" *		
IN	Preposition/subordinating conjunction	VBN	Verb, past participle
		VBP	Verb, non-3rd person singular present
JJ	Adjective		
JJR	Adjective, comparative	VBZ	Verb, 3rd person singular present
JJS	Adjective, superlative		
LS	List item marker	WDT	Wh-determiner
MD	Modal	WP	Wh-pronoun
NN	Noun, singular or mass	WP$	Possessive wh-pronoun
NNS	Noun, plural	WRB	Wh-adverb
NP	Proper noun, singular	$	Dollar Sign ($))
NPS	Proper noun, plural	#	Pound sign (#)
PDT	Predeterminer	"	Left quote (' or ")
POS	Possessive ending	"	(Right quote (" or ")
PRP	Personal pronoun	(	Left parenthesis ([, (, {)
PP$	Possessive pronoun	)	Right parenthesis (],), })
RB	Adverb	,	Comma
RBR	Adverb, comparative	.	Sentence final punc (. ! ?)
RBS	Adverb, superlative	:	Mid-sentence punc (: ; … -)
RP	Particle		

Table 8. Penn Treebank Tagset (From [20]) *Note: BES and HVS tags were not used in WSJ, but were used in Switchboard

All corpora were tagged with the Penn Treebank tag set shown in Table 8. Although the posts were also tagged using the Penn Treebank tag set and associated tagging guidelines [20], we had to make several decisions during the process that were unique to the chat domain. The first class of decisions regarded the tagging of abbreviations such as "LOL" (Laughing Out Loud) and emoticons such as ":-)" (a "smiley face" rotated on its side) frequently encountered in chat. Since these expressions conveyed emotion, we treated them as individual tokens and tagged them as "UH" (interjections).

The second class of decisions involved the tagging of sequences of non-alphanumeric characters that were not emoticons, but served a specific (formal or informal) purpose within the domain. First, based on the way the sessions were collected, user commands to both the chat room system and chatbots as well as information provided by the system and chatbots were often preceded by either the token "." or "!". Since these do not function as the Penn Treebank tag "." (sentence final punctuation), we instead tagged them as "SYM". Second, we tagged variants of tokens representing pointers such as "<--" and "^" as "PRP" (personal pronoun), since they were used to indicate a particular user (often, the user making the post itself). Finally, we tagged the token "/" as "CC" (coordinating conjunction), since it was often used in place of traditional conjunctions such as "and" and "or".

The third class involved words that, although would be considered misspelled by traditional written English standards, were so frequently encountered within the chat domain that we treated them as correctly spelled words and tagged them according to the closest corresponding word class. As an example, the token "hafta" (when referring to "have to"), if treated as a misspelling, might be tagged as "^VBP^TO", with the "^" referring to a misspelling and "VBP" and "TO" referring to "verb, non-3rd person singular present" and the word "to", respectively. However, since it was so frequently encountered in the chat domain, we often tagged it as "VBP" based on its usage. Appendix B contains a list of such words encountered in the privacy-masked version of the corpus along with their corresponding tag(s).

The final class of decisions involved words that were just plain misspelled. In this case, we tagged those words with the misspelled version of the tag. As an example, we tagged "intersting" (when referring to "interesting") as "^JJ", a misspelled adjective.

In conjunction with part-of-speech tagging, we classified each chat post in the privacy-masked corpus with a dialog act. We now turn to the details associated with this activity.

4. Chat Dialog Act Classification

We labeled each of the 10,567 privacy-masked posts using Wu et al's 15 post act categories [18], many of which were derived in part from the Stolcke et al tag set [17]. The chat dialog act classification categories as well as an example of each taken from the privacy-masked corpus are shown in Table 9.

Classification	Example
Accept	yeah it does, they all do
Bye	night ya'all.
Clarify	i meant to write the word may.....
Continuer	and thought I'd share
Emotion	lol
Emphasis	Ok I'm gonna put it up ONE MORE TIME 10-19-30sUser37
Greet	hiya 10-19-40sUser43 hug
No Answer	no I had a roomate who did though
Other	sdfjsdfjlf
Reject	u r not on meds
Statement	well i thought you and I will end up together :-(
System	JOIN
Wh-Question	11-08-20sUser70 why do you feel that way?
Yes Answer	why yes I do 10-19-40sUser24, lol
Yes/No Question	cant we all just get along

Table 9. Post Dialog Act Classification Examples

These examples highlight the complexity of the task at hand. First, we should note that we classified posts into only one of the 15 categories. At times, more than one category might apply. In addition, the *Wh-Question* example does not start with a wh-word, while the *Yes Answer* does start with a wh-word. Also, notice that the *Yes/No Question* does not include a question mark. Finally, the *Statement* example contains a token that conveys an emotion, ":-(". Taken together, these examples highlight the fact that more than just simple regular expression matching is required to classify these posts accurately. The specific interpretations we used for each chat dialog act class now follow.

Statement chat dialog acts predominantly include descriptive, narrative, and personal statements (Statements as defined by Stolcke et al) as well as other directed opinion statements (Stolcke et al's Opinion) [17]. In reality, though, *Statement* is a catch-all category, and includes other dialog act forms not covered by the other 14 chat dialog acts.

System chat dialog acts, as originally defined by Wu et al, referred to posts generated by the chat room software [18]. We expanded the notion of the system dialog act to include commands made by the user to both the chat room system as well as to personal chatbots. Finally, we also classified chatbot responses as system dialog acts.

The *Yes/No Question* chat dialog act is simply a question that can have "yes" or "no" as an answer. Similarly, the *Wh-Question* chat dialog act is a question that includes a wh-word (who, what, where, when, why, how, which) as the argument of the verb. Both correspond to Stolcke et al's Yes-No Question and Wh-Question categories, respectively [17]. However, both categories also include some dialog acts that Stolcke et al would define as Declarative, Back Channel, Open, and Rhetorical Questions.

As in Stolcke et al's definition for the category, *Yes Answer* chat dialog acts include variations on the word "yes", when acting as an answer to a *Yes/No Question*. *No Answers* are similarly defined [17].

Accept and *Reject* chat dialog acts, as in Stolcke et al's definition, all mark the degree to which the poster accepts some previous statement or opinion [17].

Greet and *Bye* chat dialog acts are defined as their name implies, and conform to Stolcke et al's Conventional Opening and Closing categories [17].

We interpreted *Clarify* chat dialog acts as posts that refer to an earlier ambiguous or unintelligible post made by the same user. As such, *Clarify* dialog acts serve to clarify the earlier post's meaning.

Continuer chat dialog acts serve to continue an earlier post of the current poster, and as such often correspond to Zitzen and Stein's split turn phenomena as described in Chapter II [5].

As the name implies, Wu et al's *Emotion* chat dialog acts express the poster's feelings, and are often characterized by the words that make the chat domain unique from tradition written domains, to include emoticons like ":-)" as well as chat abbreviations like "LOL" [18].

Emphasis chat dialog acts are used by the poster when they want to emphasize a particular point. As defined by Wu, they include the use of "really" to emphasize a verb, but also include the use of all caps as well as exclamation points [18].

Finally, the *Other* chat dialog act was reserved for posts where we could make no clear dialog act interpretation of any kind for the post.

We will now turn to the process we used to assign chat dialog act labels and part-of-speech tags to the privacy-masked chat corpus.

5. Bootstrapping Process

With the labeling guidelines decided upon, we next labeled all 10,567 tokenized posts with their corresponding part-of-speech tags and dialog act classes via a bootstrapping process. Rather than hand-tagging each individual post, we crafted a POS tagger trained on the Penn Treebank corpora, and combined with a regular expression that identified privacy–masked user names and emoticons, automatically tagged 3,507 tokenized posts. We discuss the details on the tagger approach in Chapter III, Section C. Similarly, we used simple regular expression matching to assign an initial chat dialog act to each post. We then hand-verified each token's tag within a post and as necessary changed it to its "correct" tag. Similarly, we hand-verified each post's dialog act classification and as necessary changed it to its "correct" label.

We then used the newly hand-tagged chat data, along with the Treebank corpora, to train a new tagger that automatically tagged the remaining 7,060 posts. Similarly, we used a back-propagation neural network trained on 21 features of the dialog-act labeled posts to automatically classify the remaining 7,060 posts. We discuss details of the neural network approach in Chapter III, Section C. Again, we hand-verified and as necessary corrected each token's tag (or post's dialog act label) in the new data set.

Ultimately, we annotated a total of 10,567 privacy-masked posts, representing 45,068 tokens, with part-of-speech and dialog act information.

It is important to note that we did not perform an inter-annotator agreement assessment on either the part-of-speech tags or chat dialog act classifications. This is because only one person (the author) performed the hand-verification task described earlier. As such, if the privacy-masked corpus is to be expanded further in the future, we highly recommended multiple annotators participate so that an inter-annotator assessment can be performed.

With our discussion of the corpus generation methodology complete, we now turn to a description of the machine leaning methods we used to automatically assign part-of-speech and dialog act information.

B. CHAT PART-OF-SPEECH TAGGING METHODOLOGY

Before discussing the specific part-of-speech tagger experiments we performed, it is first necessary to provide a brief overview of their mathematical foundations. The machine-learning approaches we investigated for part-of-speech tagging can be grouped into three categories: 1) Lexical n-gram taggers employing back off; 2) Hidden Markov Model taggers; and 3) Brill taggers. Our specific implementation of these approaches made use of the corresponding modules provided in the Natural Language Toolkit distribution for the Python programming language [21]. We now discuss each approach in turn.

1. Lexicalized N-Grams with Back off

The foundation for all lexicalized n-gram tagging approaches is the Markov assumption, i.e. we can predict the probability of the current event based on looking at what has happened not too far in the past. As a simplified example, let us consider a Major League Baseball player. One can make a fairly accurate prediction on the chance he will get a hit at his current at bat based on his batting performance over the immediately preceding few games. One does not necessarily make a better prediction knowing his batting performance for the entire season, or even his entire career. Of

course, there are many other variables involved in this example, e.g. the pitcher he is facing, his current health, etc. Nevertheless, this is the essence of the Markov assumption, and is used in lexical n-gram tagging models where n stands for how many words (minus one) to look into the past to help make a tagging decision.

A general discussion of lexicalized n-gram taggers can be found in [14] and [22]. Lexicalized n-grams formed the foundation for our basic tagger configuration which involved training a bigram (n = 2) tagger on a POS-tagged training set, backing off to a similarly trained unigram (n = 1) tagger, backing off to the maximum likelihood estimate (MLE) tag for the training set. Throughout the remainder of this thesis we will subsequently refer to this approach as the bigram back off tagger.

Working backwards, the MLE tag is the most common tag within a training set, and is given by

$$t_{MLE} = \arg\max_{t \in tagSet} \left[\text{count}(t) \right]$$

A unigram tagger assigns the most probable POS tag to the i^{th} word in a sequence based on its occurrence in the training data.

$$\hat{t} = \arg\max_{t \in tagSet} \left[P\left(t \mid w_i\right) \right]$$

Finally, a bigram tagger assigns the most probable POS tag to the i^{th} word in a sequence not only based on the current word, but also the previous word as well as the previous word's POS tag.

$$\hat{t} = \arg\max_{t \in tagSet} \left[P\left(t \mid w_i, t_{i-1}, w_{i-1}\right) \right]$$

Thus, our tagging approach works as follows: The tagger will first attempt to use bigram information from the training set. If no such bigram information exists, it will then back off to unigram information from the training set. If no such unigram information exists, it will finally back off to the MLE tag for the training set. The general approach is illustrated in Figure 1.

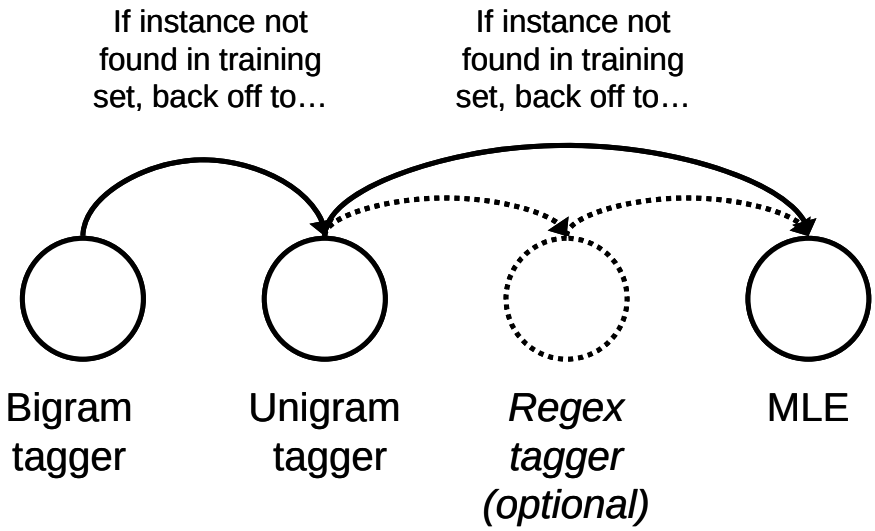

Figure 1. Bigram Back Off Tagger Approach

In addition to the basic bigram back off approach, we investigated a number of variants. The first variant, also illustrated in the previous figure, is incorporating a regular expression tagger to tag unseen instances of words via a set of regular expressions prior to backing off to the training set's MLE. For example, in the privacy-masked version of the chat corpus, all users are referred to by a standard convention, specifically, the name of the session file, followed by "User", and finally followed by a number representing when they joined the chat session. A simple regular expression can catch this naming convention, and thus correctly tag a user's privacy-masked name as "NNP" (proper noun, singular) in the event it was never observed in the training set. Of course, this can be expanded, e.g. using regular expressions to capture an unseen web address (and tag it as "NNP"), an unseen word ending in "ing" (and tag it as "VBG", or gerund verb), and so on.

The second variant we implemented involved training a bigram back off tagger on two different domains, for example, chat and the Penn Treebank. One way to accomplish this is to train the various n-gram segments of the tagger on both domains at the same time. However, if the training sets of the domains are of significantly different sizes (which is certainly the case with chat and the Penn Treebank), then either the larger domain must be sampled from to ensure it is the same size as the smaller (not preferred), or the smaller domain must be "multiplied" so that it is the same size as the larger.

Alternatively, one can "chain" two bigram back off taggers together, with each bigram back off tagger trained on a single domain. This approach is illustrated in Figure 2. In the end, we investigated both multi-domain training approaches.

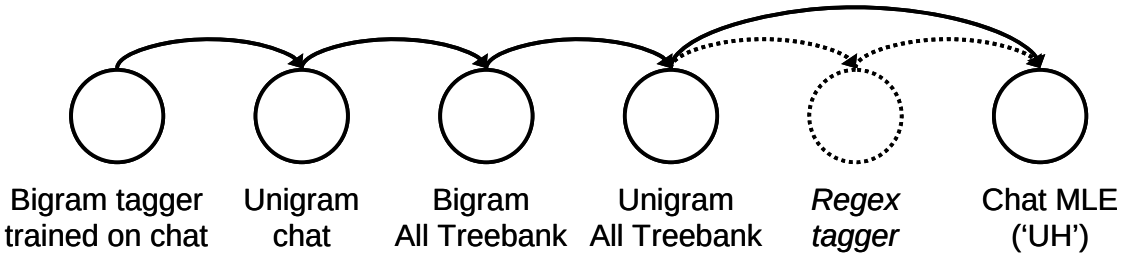

Figure 2. Multi-Domain Bigram Back Off Tagger Example

With our discussion of the bigram back off tagger completer, we now turn to a more sophisticated tagging approach, which uses a Hidden Markov Model to make its tagging decisions.

2. Hidden Markov Models

As discussed in the previous section, bigram taggers take advantage of a word's context (the preceding word and its part-of-speech tag) to assign its part-of-speech tag. Hidden Markov Model- (HMM-) based taggers take this notion of context one step further by attempting to choose the best tags for an entire sequence of words. HMMs have been applied to a number of natural language processing tasks, including speech recognition, dialog act classification, and part-of-speech tagging. Nice overviews of HMMs are provided in [13] and [14]. Our brief overview of HMMs and their application to decoding follows that of Manning and Schütze [13].

An HMM is specified by a five-tuple *(T, W, A, B, Π)*, where *T* and *W* are a set of states and output alphabet, and *Π*, *A*, and *B* are the probabilities for the initial state, state transitions, and symbol (from the output alphabet) emissions. As mentioned previously, our task is, given an observation sequence of symbols *O* and a language model $\mu = (A, B, \Pi)$ (obtained from the training set), determine the most probable sequence of states *X* that generated *O*. We can represent the space of potential state sequences with a lattice, which in this case is a two dimensional array of states versus time. Thus, we can

compute the probabilities of being at each state at each time in terms of the probabilities for being in each state at a preceding time. The Viterbi algorithm uses dynamic programming to calculate the most probable path through the lattice, and is presented in Figure 3.

ALGORITHM VITERBI$(O, HMM = (T, W, A, B, \Pi))$

Notation:

Set of states:	$T = \{t_1, t_2, ...t_N\}$
Output alphabet:	$W = \{w_1, w_2, ...w_N\}$
Initial state probabilities:	$\Pi = \{\pi_i\}, i \in T$
State transition probabilities:	$A = \{a_{ij}\}, i, j \in T$
Symbol emission probabilities:	$B = \{b_{ijk}\}, i, j \in T, k \in W$
Language model:	$\mu = (A, B, \Pi)$
State sequence:	$X = (X_1, ...X_{S+1})\ \ X_s : T \to \{1, ...N\}$
Output sequence:	$O = (o_1, ...o_S)\ \ o_s \in W$

Find: The most probable state sequence, $\hat{X} = \arg\max_X P(X \mid O, \mu)$

To do this, it is sufficient to maximize over a fixed observation sequence

$$\hat{X} = \arg\max_X P(X, O \mid \mu)$$

Define: $\delta_j(s) \leftarrow \max_{X_1 ... X_{s-1}} P(X_1 ... X_{s-1}, o_1 ... o_{s-1}, X_s = j \mid \mu)$

$\delta_j(s)$ stores for each point in the lattice the probability of the most probable path that leads to that node. The corresponding variable $\psi_j(s)$ then records the node of the incoming arc that led to this most probable path.

1. Initialization
   $\delta_j(1) \leftarrow \pi_j, \ \ 1 \le j \le N$
2. Induction
   $\delta_j(s+1) \leftarrow \max_{1 \le i \le N} \left(\delta_i(s) a_{ij} b_{ijo_s}\right), \ \ 1 \le j \le N$
3. Termination and path readout (by backtracking). The most likely state sequence is worked out from the right backwards
   $\hat{X}_{S+1} = \arg\max_{1 \le i \le N} \delta_i(S+1)$
   $\hat{X}_s = \psi_{\hat{X}_{s+1}}(s+1)$

$P(\hat{X}) = \max_{1 \le i \le N} \delta_i(S+1)$

Return $\hat{X}$

Figure 3. Viterbi Algorithm for Hidden Markov Model Decoding (After [13])

For the part-of-speech tagging problem, the task is to determine the most probable part-of-speech tag sequence (the state sequence *X*) based on the word sequence (the observation sequence *O*). Our training set of tagged data permit us to determine he language model's initial state, transition state, and symbol emission probability distributions. By working through the Viterbi algorithm, we can find the most probable tag sequence given the word sequence.

Unlike our bigram back off tagger approach, where we ultimately handled an unseen token by tagging it as the MLE for the corpus, for the Hidden Markov Model we prefer not to deal with zero counts. Unseen tokens in the training set pose two issues. First, it underestimates the actual probability, since it is always possible that our training set did not include an example. Second, though, this term will come to dominate the overall classification, since this one unseen feature value will require multiplying all other conditional probability terms with zero.

To avoid this problem, we can smooth the probability distributions used in the Hidden Markov Model tagger language model, assigning a fraction of the observed words' probability mass to the unseen words and thus providing a better estimate of the true probability distributions. A variety of smoothing approaches are available; descriptions, advantages, and disadvantages can be found in [14], [13], and [22]. For our Hidden Markov Model tagger, we decided to use the most basic approach, Laplacian smoothing, which is described next.

Laplacian smoothing, also known as Add-One smoothing, adds one to the count for unseen instances (in our case, word tokens) in the training set, redistributing the probability mass by dividing by both the total number of tokens *N* along with the total number of word types, or "vocabulary size" *V*

$$p(w_i) = \frac{c_i + 1}{N + V}$$

With our discussion of the mathematical foundation for the HMM tagger complete, we now turn to the final type of tagger evaluated, know as Brill's Transformational-Based Learning tagger.

3. Brill Transformational-Based Learning Tagging

Brill's Transformational-Based Learning tagger, here after referred to as the Brill tagger, relies on rules to determine what tags should be assigned to what words. Those rules are learned based on their usefulness when applied to a training set. Overviews of this approach can be found in [13], [14], [15], and [22]. Our presentation follows that of Brill [15].

The training set for a Brill tagger consists of a tagged corpus as well as baseline tagger. The baseline tagger can be as simple as a unigram tagger, i.e. assigning a word its most frequent tag from the tagged corpus. The Brill learning algorithm then constructs a set of tagging transformations, or rules, and employs them in order. Specifically, it employs the rule that applies to the most cases, then chooses a more specific rule that updates a fewer number of tags, and so on. As the rules get more and more specific, they may end up changing the tags of words that had already been changed by a previous rule. In essence, the Brill tagger makes an initial set of educated guesses, and then goes back and fixes any mistakes made earlier.

The possible transformations are based on a set of templates, which the Brill tagger evaluates for every possible combination, and applies those that correct the most errors. Learning stops when no more transformations can be found that can reduce the error based on a given threshold. For the nonlexicalized version of Brill's tagger, the transformation templates depicted in Table 10 are available.

Change tag *a* to *b* when:
1. The preceding (following) word is tagged *z*.
2. The word two before (after) is tagged *z*.
3. One of the two preceding (following) words is tagged *z*.
4. One of the three preceding (following) words is tagged *z*.
5. The preceding word is tagged z and the following word is tagged *w*.
6. The preceding (following) word is tagged *z* and the word two before (after) is tagged *w*.
where *a*, *b*, *z*, and *w* are variables over the part-of-speech tag set.

Table 10. Nonlexical Templates for Part-of-speech Tagging (From [15])

With these templates, the transformation learning algorithm shown in Figure 4 is as follows.

ALGORITHM TRANSFORMATIONLEARNING(*initialTagger*, *templates*, *trainingCorpus*)

1. Apply initialTagger to trainingCorpus
2. **While** transformations can still be found, **Do**
   For fromTag = tag_1 **to** tag_n

      For toTag = tag_1 **to** tag_n

         For trainingCorpus.position = 1 **to** trainingCorpus.size

            If correctTag(trainingCorpus.position)==toTag ∧
               currentTag(trainingCorpus.position)==fromTag
                  numGoodTransformations(tag(trainingCorpus.position-1))++

            Else If correctTag(trainingCorpus.position)==fromTag ∧
                  currentTag(trainingCorpus.position)==fromTag
                        numBadTransformations(tag(trainingCorpus.position-1))++

         find max_T (=numGoodTransformations(T)- numBadTransformations(T))

         If this is the best scoring rule found yet **Then** store as best rule:
            Change tag from fromTag to toTag if previous tag is T

   Apply best rule to trainingCorpus

   Append best rule to ordered list of transformations

Figure 4. Transformation Learning Algorithm for Brill Tagging (After [15])

The Brill tagger can be extended to include lexicalized templates as well. In other words, instead of just considering changing a tag from "a" to "b" based on the surrounding tags, it can also consider the surrounding words as well. Using both lexical and nonlexical templates, the transformation learning algorithm can exploit the complex interdependencies that exist between words and tags.

Now that we have covered the various approaches we used in chat part-of-speech tagging, we now describe how we set up the experiments to assess the effectiveness of each approach.

4. Part-of-speech Tagging Experimental Approach

We divided the privacy-masked chat corpus into 30 different training/test configurations, randomly selecting 10% of the corpus (1,060 posts) to serve as the test set, and the remaining 90% to serve as the training set. Collecting 30 different training/test configuration samples from the corpus permits us to compare the performance of the different taggers based on their overall accuracy, defined as

$$\text{accuracy} = \frac{\text{Number of tokens tagged correctly}}{\text{Total number of tokens}}$$

As described above, the various taggers we investigated can be grouped into the following categories: 1) N-gram back off taggers trained on various combinations of chat and/or Penn Treebank data; 2) HMM taggers trained on chat data and/or samples from the Penn Treebank; and 3) Brill taggers with various taggers serving as input and subsequently trained with chat data and/or samples from the Penn Treebank. With our discussion of the experimental approach for part-of-speech tagging complete, we now turn to our methodology for chat dialog act classification.

C. CHAT DIALOG ACT CLASSIFICATION METHODOLOGY

As with our part-of-speech tagging discussion, before we cover the chat dialog act classification experiments, it is first necessary to provide a brief overview of their mathematical foundations. First we cover the specific features we chose to measure for each post as well as our rationale. Then we detail the two main learning approaches we used in dialog act classification: 1) Back-propagation neural networks; and 2) The Naïve Bayes classifier.

1. Feature Selection

The machine-learning algorithms we used to automatically label a post with a dialog act class required a set of features on which to base classification. In Table 11 we present the initial set of features, along with their definitions and a brief rationale on why we selected them.

Feature	Definition	Rationale
f0	Number of posts ago the poster last posted	Indicator for a Continuer act
f1	Number of posts ago the poster made a spelling error	Indicator for a Clarify act
f2	Number of posts ago that a post contained a '?' but no WRB or WP POS tag	Indicator for a Yes / No Answer acts
f3	Number of posts in the future that contained a Yes or No word	Indicator for a Yes/No Question act
f4	Number of posts ago that contained a Greet word	Indicator for a Greet act
f5	Number of posts in the future that contained a Greet word	Indicator for a Greet act
f6	Number of posts ago that a post contained a Bye word	Indicator for a Bye act
f7	Number of posts in the future that contained a Bye word	Indicator for a Bye act
f8	Number of posts ago that a post was a JOIN	Indicator for a Greet act
f9	Number of posts in the future that a post is PART	Indicator for a Bye act
f10	Total number of words in post	Longer posts may be Statements and Questions, shorter posts may be Emotions and Greets/Byes, etc.
f11	First word is a conjunction, preposition, or ellipses (POS tag of 'CC', 'IN', or ':')	Indicator for a Continuer act
f12	A word contains emotion variants such as lol, ;-), etc	Indicator for an Emotion act
f13	A word contains hello or variants	Indicator for a Greet act
f14	A word contains goodbye or variants	Indicator for a Bye act
f15	A word contains yes or variants	Indicator for Yes or Accept acts
f16	A word contains no or variants	Indicator for No or Reject acts
f17	A word POS tag is WRB or WP	Indicator for a Wh-Question act
f18	A word contains one or more '?'	Indicator for Wh- or Yes/No Question acts
f19	A word contains one or more '!' (but not a '?')	Indicator for an Emphasis act
f20	A word POS tag is 'X'	Indicator for an Other act
f21	A word is a system command (. or ! with SYM POS tag)	Indicator for a System act
f22	A word is a system word, e.g. JOIN, MODE, ACTION, etc	Indicator for a System act
f23	A word is an 'any' variant, e.g. 'anyone', 'n e', etc	Indicator for a Yes/No Question act
f24	A word is in all caps, but not a system word like JOIN	Indicator for an Emphasis act
f25	A word is an 'even' or 'mean' variant	Indicator for a Clarify act
f26	Total number of users currently in the chat room	More users may stretch out distances between adjacency pairs

Table 11. Initial Post Feature Set (27 Features)

The first ten post features (f0-f9) in the table are based on the posts surrounding it, specifically, the distance to posts with particular features, with the rationale that surrounding posts should give a hint to the nature of the post itself. For example, *Continuer* dialog acts might be more likely to follow fairly closely to when the user last posted, and *Yes/No Answers* should follow fairly closely to posts with *Yes/No Question* characteristics. Note, though, that if a particular post was not found in its vicinity, we assigned it the maximum session length in the privacy-masked chat corpus, i.e., 706 posts (all sessions ranged from 687 to 706 posts). For example, at the beginning of a session, you would not be able to find the last time a poster posted (even though they may have posted just before the session was recorded). Note that this would result in an "edge effect" at the beginning and ending of the sessions, thus decreasing the validity of some of these particular features of posts near the beginning and end of the session.

The next sixteen features (f10-f25) are based on the post itself, with many of them looking for specific patterns which should give a clue on the nature of the post. For example, *Greet* dialog acts should contain a token like "hello", while *Question* dialog acts might contain a "?" as a token.

We selected the final feature (f26, current number of users logged on) because it might help normalize the distances associated with the first ten features. Specifically, more users currently logged on might increase the distances between adjacency pairs such as *Yes/No Questions* and *Yes-* or *No Answers*.

With the initial feature set having been defined, we now turn to the machine-learning methods we implemented to support chat dialog act classification.

2. Back-Propagation Neural Networks

To test the effectiveness of classifying a post with a dialog act using the 27 features, we first investigated back-propagation neural networks. Both Mitchell [23] and Luger [24] provide excellent descriptions of artificial neural networks. For brevity, we will present a conceptual overview of neural networks as well as the back propagation training algorithm as presented by Mitchell. The reader is invited to turn to Mitchell for a derivation of the back-propagation rule itself.

The fundamental building block for all artificial neural networks is the artificial neuron, referred to hereafter as the unit. The unit takes a series of inputs (either from the environment or other units), applies a weight to each input, and based on its internal threshold function, emits an output signal. The threshold function used by the units, as well as how they are combined together, define the variety of decision surfaces that the neural network can perform. Thus the training task associated with artificial neural networks is as follows: based on a set of inputs and target outputs, learn the weights for each unit such that the total error between actual network outputs and the target outputs is minimized.

Back-propagation neural networks combine multiple unit layers along with a differentiable threshold function for each unit, permitting a rich variety of decision surfaces. In particular, our implementation uses, in addition to the output layer, a single hidden layer of units. Although there are a variety of sigmoid functions available to serve as a threshold function, the one we choose is the inverse tangent function, *arctan(x)*. This particular function has the (computationally) useful property that its derivative is easily expressed as the function itself, namely,

$$\frac{d\arctan(x)}{dx} = \frac{1}{1+x^2} = 1 - \left(\arctan(x)\right)^2$$

Thus, the output is a continuous function of a weighted sum of its inputs, or $o = \arctan(\vec{w} \cdot \vec{x})$, where o is the output value, $\vec{w}$ is the weight vector, and $\vec{x}$ is the input vector. With the sigmoid function now defined, we turn to its implementation in the back-propagation neural network training algorithm, presented in Figure 5.

ALGORITHM BACK-PROPAGATION $(trainingSet, \eta, n_{in}, n_{out}, n_{hidden})$

Each element of *trainingSet* is a pair of the form $(\vec{x}, \vec{t})$, where $\vec{x}$ is the vector of network input values and $\vec{t}$ is the vector of target network output values.

η is the learning rate, n_{in} is the number of network inputs, n_{hidden} is the number of hidden units, and n_{out} is the number of output units.

The input from unit *i* to unit *j* is denoted x_{ji}, and the weight from unit *i* to unit *j* is denoted w_{ji}.

1. Create a feed-forward network with n_{in} inputs, n_{hidden} hidden units, and n_{out} output units.
2. Initialize all network weights w_{ji} to small random numbers (e.g., between -0.5 and 0.5)
3. **While** termination condition not met, **Do**:

   For each $(\vec{x}, \vec{t})$ in *trainingSet,* **Do**:

   Propagate the input forward through the network

   a. Input the instance $\vec{x}$ to the network and compute the output o_u of every unit *u* in the network

   $$o_u = \arctan(\vec{w}_u \cdot \vec{x}_u)$$

   Propagate the errors backward through the network

   b. **For each** network output unit *k*, calculate its error term δ_k

   $$\delta_k \leftarrow (1 - o_k^2)(t_k - o_k)$$

   c. **For each** hidden unit *h*, calculate its error term δ_h

   $$\delta_h \leftarrow (1 - o_h^2) \sum_{k \in outputUnits} w_{kh} \delta_k$$

   d. Update each network weight w_{ji}

   $$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

   where

   $$\Delta w_{ji} = \eta \delta_j x_{ji}$$

Figure 5. Back-Propagation with Gradient Descent for Neural Network Training (After [23])

In our case, the input vector representing a particular post is the set of its features, and thus has 27 dimensions. The features themselves were normalized by their maximum value seen for a particular feature, thus restricting their range to a real number between zero and one. Similarly, the output for the neural network is a vector with a dimension equal to 15, the number of chat dialog act classes. Thus, the training set for our back-propagation neural network consists of the training posts' feature vectors and their target output vectors (with "1" assigned for the actual dialog act classification and "0" for all other classes).

To build the back-propagation network, we used Schemenauer's implementation for the Python programming language [25]. In addition to the above parameters, we used 16 hidden nodes and a learning rate of 0.05. Note that we did not perform a formal optimization to determine these values. Instead, we varied them around set values and selected the configuration that reduced the global error on a training set the most after twenty iterations on each configuration.

With the discussion of the neural network implementation complete, we now turn to the second machine-learning method we investigated, the Naïve Bayes Classifier.

3. Naïve Bayes Classifier

Manning and Schütze [13], Jurafsky and Martin [14], Mitchell [23], and Luger [24] provide nice overviews of the general Bayesian learning approach. Following Mitchell, we will first describe the Bayesian approach, show how the Naïve Bayes classifier follows from it, and then discuss how we used it with respect to the dialog act classification.

Given a training set consisting of instances with features represented by a vector $\vec{F}$ consisting of elements $f_i \in Features$ along with their associated classifications $C \in Classes$, we can calculate the probabilities of those features given their classification as well as the prior probability of the class itself. From Bayes theorem, we have

$$P\left(C \mid \vec{F}\right) = \frac{P\left(\vec{F} \mid C\right)}{P\left(\vec{F}\right)}$$

Since the denominator for each particular class is the same, we can assign the most probable class for an unseen instance by

$$\hat{C} = \arg\max_{C_i \in Classes} P\left(\vec{F} \mid C_i\right) P\left(C_i\right)$$

The Naïve Bayes classifier makes the simplifying assumption that the feature values are conditionally independent of the classification. Therefore, the probability of observing $f_1 \wedge f_2 \wedge \ldots \wedge f_n$ given a class C_i is just the product of the probabilities of the individual features given the class, $P\left(f_1 \mid C_i\right) P\left(f_2 \mid C_i\right) \ldots P\left(f_n \mid C_i\right)$. Substituting this in the general Bayesian learning approach gives us the Naïve Bayes Classifier

$$\hat{C} = \arg\max_{C_i \in Classes} P\left(C_i\right) \prod_j P\left(f_j \mid C_i\right)$$

As with the Hidden Markov models employed in part-of-speech tagging discussed earlier, we must account for the possibility that our training set contains zero counts for a particular feature. There, we smoothed using the Laplacian estimate of the probability. However, we found for the Naïve Bayes classifier for chat dialog acts that the Witten-Bell probability estimate worked well, and thus we briefly describe its use next.

The key behind many smoothing approaches is to estimate the counts of things never seen by the counts of things seen once. For Witten-Bell (described in [14]), the probability mass reserved for unseen events is equal to $T/N+T$ where T is the number of observed event types and N is the total number of observed events. This equates to the maximum likelihood estimate of a new type event occurring. The remaining probability mass is discounted such that all probability estimates sum to one, yielding

$$\begin{aligned} p(f_i) &= T/Z\left(N+T\right) \quad \text{if } \mathrm{c_i} = 0 \\ &= c_i/\left(N+T\right) \quad \text{if } \mathrm{c_i} \neq 0 \end{aligned}$$

where f_i is a particular feature, c_i is its count in the training set, and Z is the total number of events with zero count, or

$$Z = \sum_{i:c_i=0} 1$$

With the Naïve Bayes classifier and Witten-Bell smoothing discussion complete, we can now describe how we used it to automatically assign the dialog act class for a

particular post. Given a training set of posts, with each post containing 27 feature values as well as a dialog act class, we calculated both the prior class probability distributions as well as the conditional probability distributions for each feature given a class. We then smoothed these distributions by the total possible values for each particular feature. Finally, we used these smoothed distributions in the Naïve Bayes classifier to automatically assign the class for an unseen instance in the test set of posts.

Now that we have covered both of the machine-learning approaches used in chat dialog act classification, we now describe our experimental set-up to assess the effectiveness of each approach.

4. Chat Dialog Act Classification Experimental Approach

As with the part-of-speech tagging experiments, we divided the privacy-masked chat corpus into 30 different training/test configurations, randomly selecting 10% of the corpus (1,060 posts) to serve as the test set, and the remaining 90% to serve as the training set. Collecting 30 different training/test configuration samples from the corpus permits us to compare the performance of the different learning approaches based on the mean and standard deviation of several different performance scores.

The first performance score we measured for each training/test configuration was the overall accuracy of the learning method. Similar to part-of-speech tagging, accuracy is defined as

$$\text{accuracy} = \frac{\text{Number of posts labeled correctly}}{\text{Total number of posts}}$$

Unlike the part-of-speech tagging situation, the number of classification labels is relatively small. As such, we found it particularly insightful to calculate both recall and precision scores for each class in each training/test configuration. Their definitions are as follows.

$$\text{recall} = \frac{\text{Number in class labeled correctly}}{\text{Actual number in the class}}$$

$$\text{precision} = \frac{\text{Number in class labeled correctly}}{\text{Total number labeled as the class}}$$

Finally, although recall and precision enable us to assess each learning method's performance at the dialog act classification level, it is useful to have a single measure for its performance. The harmonic mean of the precision and recall scores, known as the f-score, is a good measurement because it does not permit improving one aspect of performance at the expense of the other. As such, f-score is defined as

$$\text{f-score} = \frac{2}{1/\text{precision} + 1/\text{recall}}$$

With our description of the experiment complete, we are ready to compare the performance of the back-propagation neural network and Naïve Bayes machine-learning approaches. The results of these experiments along with those of the part-of-speech taggers are presented in Chapter IV.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. TESTING AND ANALYSIS

In this chapter we present the results of our experiments as well as provide a discussion on their significance. We will first cover some general statistics of the privacy-masked corpus we collected, and provide comparisons to other language domains of similar size. We will then review the performance of the various machine-learning approaches we used for both part-of-speech tagging and chat dialog act classification.

A. CORPUS STATISTICAL COMPARISON

Before trying to build a highly accurate tagger for the chat domain, we first needed to compare the chat domain to some baseline in order to assess the potential for tagger performance. Since we had 10,567 tagged chat posts, we were initially inclined to select training/test sets consisting of 10,567 sentences from the other domains. However, the unit of concern is at the token-, and not sentence-level. Therefore, this would be inappropriate, since both Wall Street Journal and Switchboard sentences were on average much longer than chat posts. Since the 10,567 tagged chat posts contained 45,068 tokens, we randomly selected 30 different contiguous sections from both the Wall Street Journal and Switchboard corpora, with each sample containing the same number of tokens as the privacy-masked corpus (plus those necessary to complete the last sentence) to serve as source data for those domains.

We then measured a number of lexical statistics on the chat privacy-masked corpus as well as the Wall Street Journal and Switchboard corpora samples. In particular, we measured the token/type, part-of-speech (POS) tag count/type, and POS tag count/token ratios. The token/type ratio is defined as the total number of words (tokens) in the corpus sample divided by the total number of unique words (types). The POS tag count/type ratio is defined as the average of the number of part-of-speech tags for each type in the sample. Finally, the POS tag count/token ratio is defined as the average of the number of part-of-speech tags for all tokens in the sample.

We also trained and tested unigram taggers (backing off to the domain's MLE), HMM taggers, and Brill taggers (with the aforementioned unigram taggers serving as the initial tagger) for each of the domains, using a single representative sample from each of the Wall Street Journal and Switchboard samples collected earlier. From those selections, we then created 30 different training/test sets by randomly removing 10% of the sentence-level units from each domain sample to serve as test data with the remainder serving as training data.

With this brief overview of the baseline comparison methodology complete, we can now discuss the corpora lexical statistics, summarized in Table 12.

	Privacy-masked Chat	Wall Street Journal	Switchboard
Sentence Level Units: Mean	10567	867.533	2866.000
Std Dev	-	49.907	225.960
Tokens: Mean	45068	45094.167	45074.533
St Dev	-	27.969	26.531
Types: Mean	5803	7094.033	3046.900
St Dev	-	192.415	80.910
Misspelled Tokens: Mean	490	0	36.433
St Dev	-	0	10.484
Misspelled Types: Mean	433	0	27.133
St Dev	-	0	6.745
Token/Type: Mean	7.766	6.361	14.804
St Dev	-	0.177	0.399

Table 12. Corpora Lexical Statistics Summary

1. Corpora Sample Token/Type Ratios

Since all the domain samples were roughly the same size (measured by number of tokens), the token/type ratio represents the size of each domain's vocabulary. In other words, as the token/type ratio gets larger, the vocabulary of the domain sample (measured by number of types) gets smaller, since all samples contained roughly the same number of tokens. As can be seen in Table 12, the chat token/type ratio of 7.766 is much closer

to the Wall Street Journal corpus than that of Switchboard as represented by the means and standard deviations from 30 similarly sized samples from each domain.

These findings are consistent with Freiermuth's comparative analysis of political discussion in the written, spoken, and chat domains, although his samples were much smaller (3000 tokens/domain) [9]. This finding bears further discussion. Based on the fact that a conversation is taking place, online chat may seem like spoken language. However, from a lexical perspective, it is much more diverse, and thus more closely resembles traditional written language. Apparently, one's ability to edit his/her post before pressing "Enter" allows them to be more selective in the words they choose to use. As described in Chapter II, the *Contexts of Production* and *Use* are synchronous for spoken language, thus inhibiting the participants' ability to find preferred words because they are either trying to maintain the floor or avoid silence. By contrast, the *Contexts of Production* and *Use* are asynchronous in traditional written language as well as chat, with the token/type ratio being one piece of evidence for this asynchronicity.

There are some things unique to the privacy-masked chat domain, though, that directly affect its token/type ratio, and are thus worth mentioning. First of all, the privacy masking activity itself has the effect of increasing the token/type ratio. This is because all direct references to chat participants were replaced with a single, unique name per participant. In many cases, though, chat participants were referred to by more than one name (from our example in Chapter III, "killerBlonde51", "killer", "Blondie", "kb51", etc.).

Second, the privacy-masked chat corpus (and the chat domain in general) is littered with misspellings, which will decrease the token/type ratio. Specifically, we tagged 490 tokens in the privacy-masked chat corpus as misspellings. Of these, 433 were unique misspellings. Thus, roughly one percent of the privacy-masked chat corpus contained misspellings. This is in comparison to both Wall Street Journal articles and the transcribed Switchboard spoken conversations, which contain virtually no misspellings (see Table 12).

Finally, several of the emoticons and chat abbreviations have the "property" that they can contain repetition of characters within the word. These variants of the same expression also decrease the token/type ratio for chat. For example, we observed a number of variants for the emoticon "<3" (a heart shape on its side): "<333", "<33333333", "<3's", etc. Each of these variants was counted as a separate type. We did not treat these as misspellings, and instead tagged them as interjections. Note that the same property occurred in traditional words, e.g. "reeeeeallllllly" for "really", although in these cases we did tag them as misspellings. Regardless of how they were tagged, though, these unique types, albeit with the same "root", add to the lexical diversity of the chat domain per our definition.

What effect does the chat token/type ratio have on stochastic part-of-speech tagging? As mentioned earlier, a smaller token/type ratio means a larger vocabulary for the domain. As such, a corpus with a larger token/type will generally have more data to train a part-of-speech tagger than a similarly sized corpus with a smaller token/type ratio. Regarding the special case for misspellings, it will be difficult for a stochastic tagger to correctly tag a misspelling, since its type may only occur in the corpora a few times at most (depending on its size). Thus, the token/type ratio could be a significant factor in stochastic tagger performance, but it is not the only one. In particular, the part-of-speech ambiguity for a particular word, represented overall for a corpus by its POS tag count ratios, will also play a role.

2. Corpora Sample POS Tag Count/Type Ratios

One of the measures of a word's lexical ambiguity is the number of part-of-speech tags it can have when in use. Words that have only one part-of-speech tag, for example, "the" (tagged "DT" for determiner) are unambiguous. On the other hand, if a word has more than one possible part-of-speech tag, e.g. the word "bear", the machine-learning algorithm has a decision to make. Thus, words with more than one part-of-speech tag are ambiguous, and it is these words that determine the upper limit for overall tagging accuracy. The part-of-speech tag counts for both tokens and types within the privacy-masked chat corpora are presented in Table 13 below.

Chat POS Tag Count	Chat Type Count	Chat Token Count
1	5141	20867
2	489	10175
3	121	5988
4	30	3472
5	16	3577
6 (“s”, “a”, “of”, “there”)	4	947
7 (“n”, “‘”)	2	42
Total Counts	5803	45068
POS Tag Count Ratios	1.158	2.151

Table 13. POS Tag Counts for Privacy-masked Chat Corpus Types and Tokens

As can be seen, even though the vast majority of the chat types have only one part-of-speech tag, less than half of the tokens in the privacy-masked corpus are of this variety. In particular, note that more than a quarter of the tokens have three or more part-of-speech tags. In fact, many of the types with part-of-speech tags numbered five and greater include a misspelling part-of-speech tag. Thus, since the tagger is concerned with tagging words in use (tokens), the POS tag count/token ratio (as opposed to the corresponding type ratio) will have the most impact on overall tagger performance. We present a comparison between the samples from the three domains in Table 14.

	Privacy-masked Chat	Wall Street Journal	Switchboard
POS Tag Count/Type: Mean	1.158	1.141	1.186
St Dev	-	0.006	0.008
POS Tag Count/Token: Mean	2.151	1.459	1.833
St Dev	-	0.079	0.105

Table 14. Corpora POS Tag Count Ratio Summary

As can be seen, chat has the largest POS tag count/token ratio for the three domains, with over two tags per token on average. Switchboard follows with a ratio of 1.833, with Wall Street Journal having the least part-of-speech ambiguity with a 1.459

tags/token ratio. How does this ambiguity affect impact tagger performance? As mentioned earlier, a stochastic tagger will in general have a more difficult task in selecting the correct part-of-speech the more labels it has to choose from. However, this will be offset by the amount of lexical data it has to train from, represented by the corpus's token/type ratio.

With these two measures in mind, we can now see how they might affect part-of-speech taggers trained on the same amount of data from the same domain.

3. Tagger Self Domain Comparison

The various tagger accuracies, each trained on data only from their own domain, are shown in Table 15.

	Privacy Masked Chat	Wall Street Journal Sample	Switchboard Sample
Sample Size (Tokens)	45068	45074	45085
Token/Type	7.766	6.387	14.777
POS Tag Count/Token	2.151	1.428	1.803
Unigram to MLE Accuracy: Mean	0.8123	0.8223	0.8577
Std Dev	0.0069	0.0066	0.0071
HMM Accuracy: Mean	0.8699	0.8869	0.9132
Std Dev	0.0062	0.0080	0.0049
Brill Accuracy: Mean	0.8601	0.8659	0.8998
Std Dev	0.0071	0.0069	0.0052

Table 15. Self Domain Tagger Performance Comparison

As can be seen, all part-of-speech taggers performed the best on the Switchboard corpora sample, achieving over 91% accuracy with its Hidden Markov Model tagger. It appears that, although its part-of-speech ambiguity is between the other two domains, tagger performance is assisted by the fact that the Switchboard sample has nearly twice as many tokens per type, providing more information to base its tagging decisions upon. Indeed, its unigram-MLE tagger performs nearly as well as the best performing taggers for the other domains.

The next best performing domain was the Wall Street Journal, followed surprisingly close by the privacy-masked chat corpus. Indeed, the chat domain's Brill tagger nearly equaled its counterpart for the Wall Street Journal sample. This is interesting, since although chat usage may appear to be "wild", it confirms the fact that with all communication domains, there are both lexical and syntactic rules that govern acceptable structure. This leads one to ask the question of why a domain with over one percent of its tokens misspelled (as well as a much greater part-of-speech ambiguity) can almost equal the tagging performance of a more structured (albeit complex) domain. Certainly there are other factors that play a role, not the least of which is the syntactic structure of the sentences in the domains themselves. Nevertheless, these results are encouraging, and provide a level of confidence that state-of-the-art taggers employed on chat should reach similar accuracy rates given similar amounts of training data.

With this baseline comparison complete, we now turn to presenting the results of our efforts to maximize the performance of part-of-speech taggers for the chat domain.

B. CHAT PART-OF-SPEECH TAGGING RESULTS

In this section we present the results of our part-of-speech tagging experiments. We will first present the accuracy of various N-gram back off taggers, followed by the HMM taggers, and finally the Brill taggers. Throughout, we will provide comments on both the effectiveness and significance of the various tagging approaches.

1. N-Gram Back Off Tagger Performance

In our discussion on the n-gram tagger performance, we will first review the performance of the taggers each trained on the Wall Street Journal, Brown, Switchboard, the entire Penn Treebank, and Chat domains. We then cover the n-gram taggers trained on combinations of those domains, to include some performance enhancements over the basic n-gram back off approach.

a. N-Gram Back Off Trained on Single Domain

The mean accuracy and associated standard deviation for unigram and bigram taggers trained on a single domain and tested on the chat domain are shown in Table 16, and graphically in Figure 6. Note that error bars in all subsequent tagger accuracy plots represent +/-1 standard deviation for the mean accuracy figure.

	Accuracy: Mean	Accuracy: St Dev
Switchboard Unigram	0.5329	0.0085
Bigram	0.5387	0.0082
WSJ Unigram	0.5396	0.0082
Bigram	0.5505	0.0080
Brown Unigram	0.5460	0.0089
Bigram	0.5587	0.0091
All Treebank Unigram	0.5877	0.0078
Bigram	0.6006	0.0078
Chat Unigram	0.8123	0.0069
Bigram	0.8242	0.0074

Table 16. N-Gram Back Off Tagger Performance on Chat Trained on a Single Domain

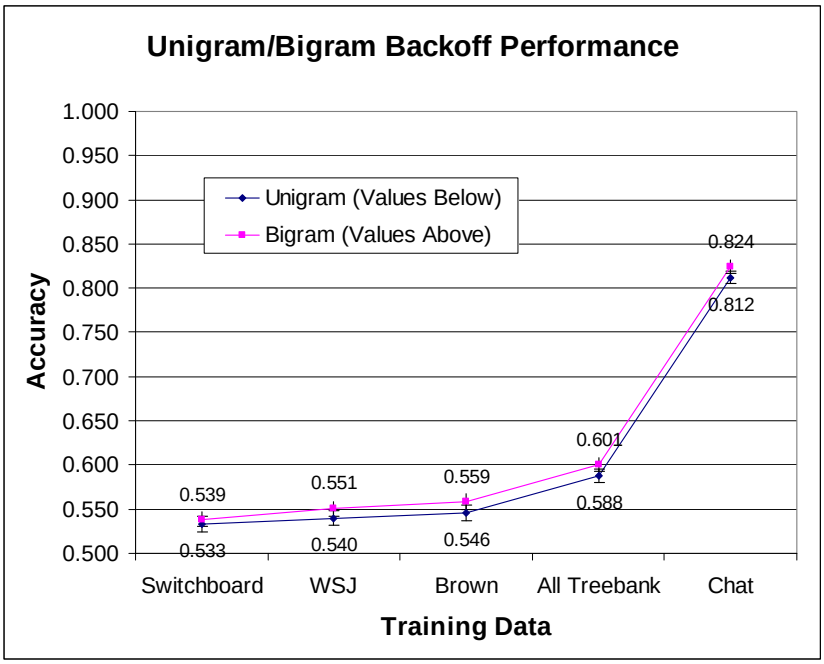

Figure 6. N-Gram Back Off Tagger Performance on Chat Trained on a Single Domain

Recall that for the unigram taggers, a tag is assigned based on the word type's most prevalent tag in the training data. If no instance is found in the training data, the tagger backs off to the most prevalent tag in the entire domain, referred to as the maximum likelihood estimate (MLE). The MLE for the Wall Street Journal, Brown, and the entire Treebank (which also includes Switchboard) is "NN"; the MLE for Switchboard alone is ","; finally, the MLE for the privacy-masked chat corpus is "UH". The approach is the same for the bigram tagger, except that they first use a bigram instance (if the training data contains it) before backing off the unigram and ultimately the domain MLE.

Several things are readily evident in Figure 6. Notice first that the accuracies of the bigram taggers are only marginally better than their unigram counterparts trained on a given domain. Also, notice that there is little difference between the accuracies of taggers (54-55%) trained on only one corpus from the Treebank. However, when all Treebank corpora are included in the training set, the accuracy jumps up to 60.1% for the bigram back off tagger version.

Although not surprising, it is nonetheless striking to see the performance improvement when the tagger is trained on chat data. Relatively few words are required from the chat domain (~41,000 training set tokens) to get 82.4% accuracy using the bigram back off tagging approach alone. This is compared to 60.1% when training on millions of words from the written and spoken domains, as represented by the Penn Treebank. This brings home a fundamental point of our work. At least from a vocabulary perspective, the chat domain is fundamentally different than that of either traditional written or spoken domains. That being said, we seek to understand whether those domains are still of some benefit from both a lexical as well as syntactical perspective to provide tagging performance improvements over methods using only a small amount of training data (albeit exactly the right kind of training data). Our next section will start to address this issue.

b. N-Gram Back Off Tagger Performance Improvements

Performance improvements over the back off taggers discussed in the previous are shown in Table 17, and graphically in Figure 7.

	Accuracy: Mean	Accuracy: St Dev
Chat to Switchboard: Chained Unigram	0.8464	0.0052
Chained Bigram	0.8612	0.0053
Chat to WSJ: Chained Unigram	0.8508	0.0050
Chained Bigram	0.8647	0.0051
Chat to Brown: Chained Unigram	0.8542	0.0054
Chained Bigram	0.8685	0.0054
Chat to All Treebank: Chained Unigram	0.8604	0.0046
Chained Bigram	0.8761	0.0045
Chained Bigram w/ Regex (Chat + Treebank)	0.8917	0.0043
Combined Corpora Bigram w/ Regex (Chat + Treebank)	0.8984	0.0045

Table 17. N-Gram Back Off Tagger Performance Improvements

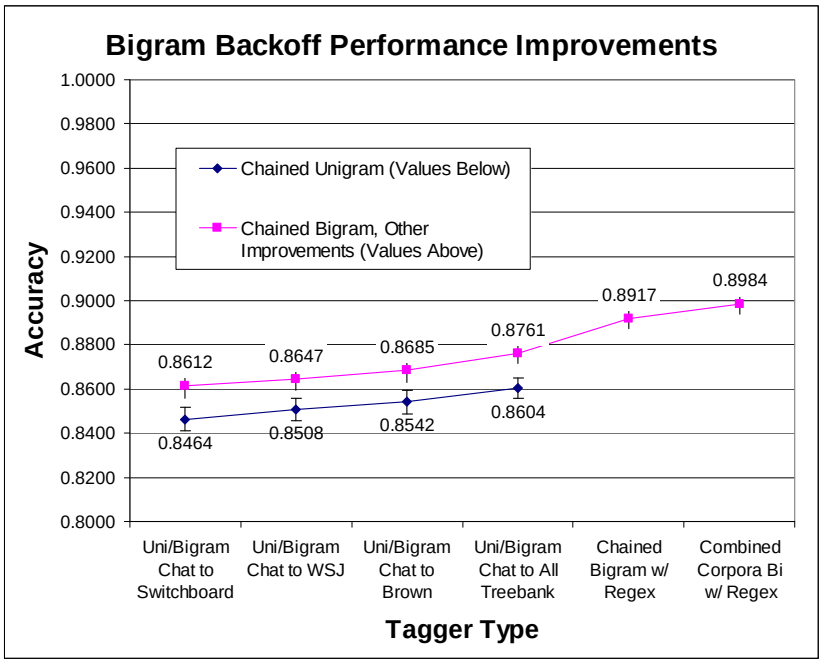

Figure 7. N-Gram Back Off Tagger Performance Improvements

Recall that the chained unigram (bigram) back off tagger incorporates a unigram (bigram) back off tagger trained first on chat. However, instead of backing off immediately to the chat MLE, the tagger first backs off to another unigram (bigram) back off tagger trained on another domain. Unlike the case for unigram / bigram back off taggers trained on a single domain, however, there appears to be a significant improvement in performance using the bigram information. Regardless, incorporating multiple domains as part of the training set provide significant improvement to the bigram back off tagger trained on chat alone.

The final two taggers bear some explanation. The first is a chained bigram back off tagger trained on both chat and the entire Penn Treebank. However, before backing off to the chat MLE, it first uses a regular expression that recognizes privacy-masked names and tags them as "NNP". More importantly, though, it uses standard morphological rules (e.g., adverbs end in "ly", plural nouns end in "s", etc.) to assign a likely tag. Incorporating this regular expression provides a significant 1.5% improvement in total accuracy over the same tagger not using the regular expression.

The final tagger also uses the same regular expression. However, instead of using multiple domains via a chaining approach, it instead trains on all the corpora at the same time, resulting in a single bigram back off tagger. Since the chat training data is much smaller (thousands of words as opposed to millions of Treebank words), it must be "multiplied" so that its effect is not drowned out by the larger Treebank data set. Through an informal optimization, we determined that multiplying chat by 70 resulted in the best accuracy improvement. Overall accuracy for this approach is 89.8%.

Although adding additional domains clearly improves the bigram back off tagger performance, the tagging algorithm itself is relatively simple. As such, performance improvement can largely be attributed to the additional vocabulary provided by the Penn Treebank corpora. Of course, we want to use this additional information, in conjunction with more sophisticated tagging approaches, to improve tagging accuracy even more. With this in mind, we turn now to the Hidden Markov Model (HMM) tagger results.

2. Hidden Markov Model Tagger Performance

Hidden Markov Model taggers, by the nature of the algorithms used, take considerably longer than the n-gram back off tagger we investigated both to train as well as to assign the most likely tag sequence given a string of tokens. As such, we took the following testing approach. First, we ran 30 different training/test sets, with each tagger trained only on the particular chat training data set. Second, we trained an HMM using samples of size ~45,000 tokens from both the Wall Street Journal and Switchboard. In the same fashion as before, we multiplied each chat training data set by seven to ensure it did not get drowned out by the addition of the other non chat data. For both the chat only and chat + WSJ/Switchboard sample configurations, we calculated the mean accuracies and standard deviations. Finally, we selected the one training/test set pair (out of 30) that had the closest accuracy to the mean accuracy of the HMM taggers trained only on chat. For this training/test pair, we trained on chat data (multiplied by varying amounts) combined with the entire Penn Treebank. The accuracies and standard deviations for the HMM tagger experiments are shown in Table 18, and graphically in Figure 8.

	Mean	St Dev
HMM Chat	0.8699	0.0062
HMM Chat X 7 + WSJ, Switchboard Samples	0.8853	0.0054
HMM Single Sample: Chat X 150 + Treebank	0.903	-

Table 18. Hidden Markov Model Tagger Performance

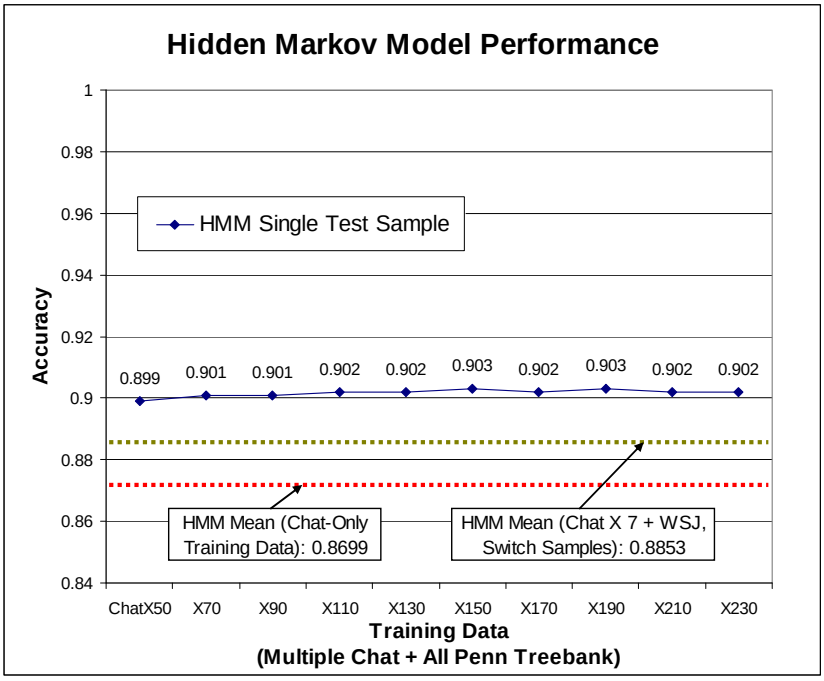

Figure 8. Hidden Markov Model Tagger Performance

First mentioned in Chapter IV Section A, the HMM tagger trained only on chat data achieves a mean accuracy of 87.0% (Table 15), a significant increase over the best bigram back off tagger trained only on chat (82.4%; see Table 16). The HMM tagger trained on chat and samples from the WSJ and Switchboard performed significantly better than an HMM tagger trained on chat alone, achieving a mean accuracy of 88.5%.

For the single training/test pair trained on both chat and the entire Treebank, we achieved a maximum accuracy of 90.3% when the chat training set was multiplied by 150. This result suggests that an HMM tagger trained on both chat and the entire Treebank might perform significantly better than the best performing tagger presented so

far, the combined corpora bigram back off tagger, which had a mean accuracy of 89.8%. Indeed, the HMM tagger could perform even better than the accuracy figures suggest, since the bigram back off tagger incorporates a regular expression that automatically tags privacy-masked user names. The HMM tagger does not rely on regular expression in assigning its most likely tag sequence, giving it a better chance at correctly tagging non-privacy-masked user names as "NNP".

3. Brill Tagger Performance

As discussed in Chapter III, there are two aspects to Brill tagger training. First, there is the training of the tagger that serves as input to the transformation learning algorithm. For the input tagger, Brill suggests using a unigram approach that tags each word with its most common part-of-speech tag [15]. Then, there is the implementation of the algorithm itself, which learns a sequence of rules that, when iteratively applied to the input tagger, improves upon its performance.

As with the other two tagging approaches, our goal is to combine chat training data with corpora from both the written and spoken domains to maximize part-of-speech tagging performance. However, it takes both a significant amount of time and memory for Brill's transformation learning algorithm to learn a reasonable number of rules (250) based on a large training set. Thus, for our initial Brill tagging experiments, we took the following approach. For the input tagger, we selected one training/test set pair (out of 30) that had the closest accuracy to the mean accuracy of the chained unigram tagger that incorporates a regular expression (87.56%). For this training/test pair, we then used the transformation learning algorithm to train on chat data (multiplied by varying amounts) combined with 50% of the Wall Street Journal. The accuracies and associated standard deviations for these Brill tagger experiments are shown in Table 19 and graphically in Figure 9.

	Mean	St Dev
Chained Unigram: Chat to All Treeebank to Regex	0.8756	0.0043
Brill Single Sample: Chat X 30 + 50% WSJ	0.8988	-

Table 19. Brill Tagger Performance for Single Chat Test Set

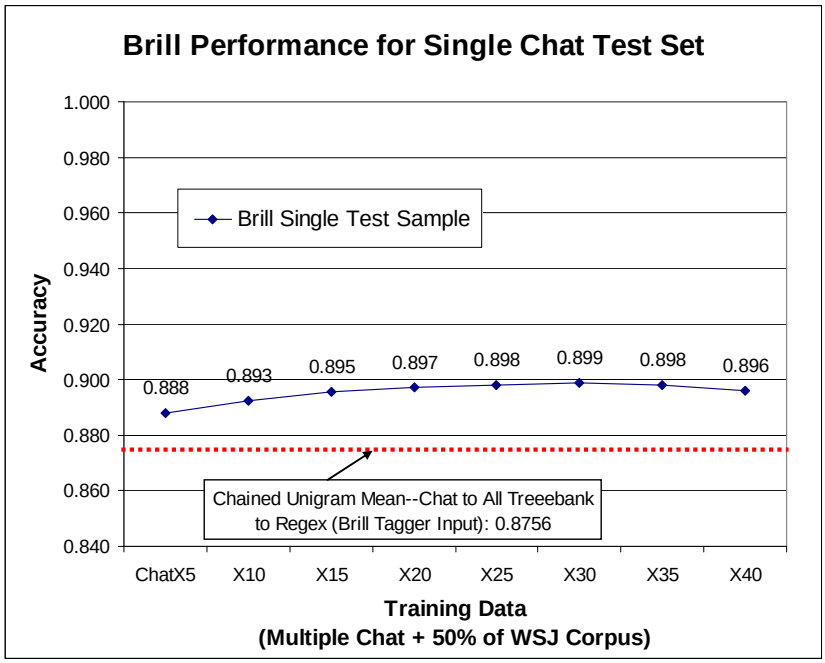

Figure 9. Brill Tagger Performance for Single Chat Test Set

For the single training/test pair using a chained unigram back off tagger subsequently learning rules based on both chat and 50% of the Wall Street Journal, we achieved a maximum accuracy of 89.9% when the chat training set was multiplied by 30. Indeed, this is significantly better than the performance of the Brill tagger trained only on chat data (first mentioned in Chapter IV Section A), with a mean accuracy of 86.0% (see Table 15). This result suggests that a Brill tagger trained on both chat and the entire Treebank might also perform significantly better than the best performing tagger presented so far, the combined corpora bigram back off tagger, which had a mean accuracy of 89.8% (see Table 17 and Figure 7).

In addition to using a chained unigram back off tagger, we investigated using our most accurate n-gram back off taggers as input into the Brill transformation learning algorithm. First, we used our chained bigram back off tagger incorporating a regular expression, with a mean accuracy of 89.2%. Second, we used the combined corpora bigram back off tagger (which included the entire Treebank plus the chat training set multiplied by 70), with a mean accuracy of 89.8%. For both of these input tagger sets, we trained with the transformation learning algorithm only on chat data. Finally, we used the combined corpora bigram back off tagger, but trained with the algorithm on both chat data (multiplied by seven) as well as using samples of size ~45,000 tokens from both the Wall Street Journal and Switchboard. The accuracies and associated standard deviations for these Brill tagger experiments are shown in Table 20, and graphically in Figure 10.

	Accuracy: Mean	Accuracy: St Dev
Chained Bigram w/ Regex (Chat + Treebank)	0.8917	0.0043
Brill Encapsulation of Chained Bigram, Trained on Chat	0.9059	0.0047
Combined Corpora Bigram w/ Regex (Chat + Treebank)	0.8984	0.0045
Brill Encapsulation of Combined Corpora Bigram, Trained on Chat	0.9069	0.0050
Brill Encapsulation of Combined Corpora Bigram, Trained on ChatX7 + WSJ, Switch samples	0.9077	0.0045

Table 20. Brill Tagger Performance Improvements

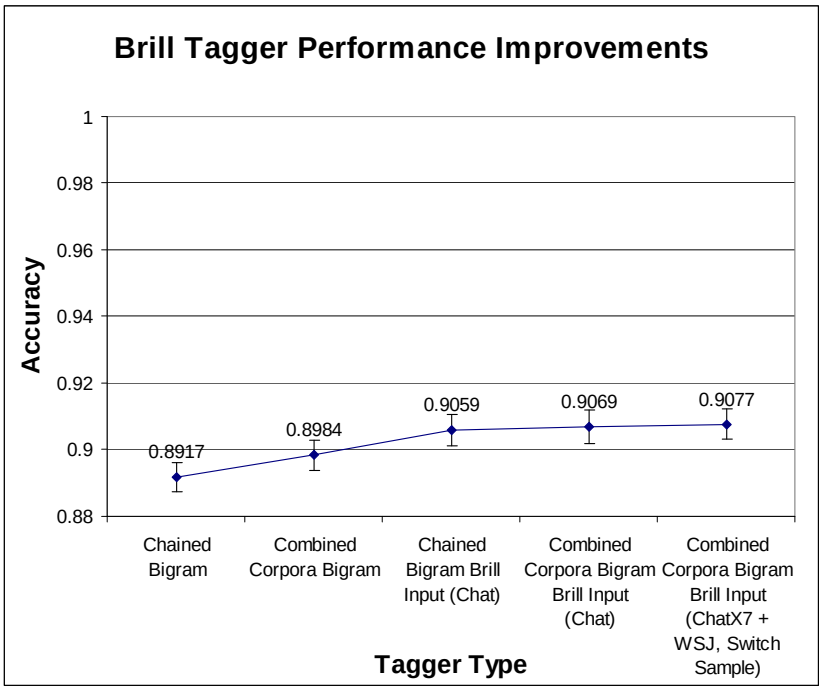

Figure 10. Brill Tagger Performance Improvements

Encapsulating the chained bigram and combined corpora bigram back off taggers with rules learned by the transformation learning algorithm result in a significant improvement in accuracy. The best performing Brill tagger achieved a mean accuracy of 90.8%. However, based on the standard deviations, this is not a significant improvement over the accuracy of any of the other Brill taggers, with accuracies of 90.6% and 90.7%. Based on the earlier Brill results, the addition of more non-chat training data for the Brill learning algorithm should improve performance. That being said, achieving Brill tagger accuracies significantly greater than 91% appears unlikely within the current privacy-masked chat corpus framework.

4. Discussion

As mentioned earlier, results from single training/test sample pairs suggest that significant performance improvements are achievable with both HMM and Brill approaches. What we did not investigate, however, was whether there was an optimal ratio of the various Treebank corpora to use to improve tagger performance on chat. Varying the amount of training data from each Treebank corpora, although it may

degrade performance of the simpler n-gram back off taggers, may actually improve performance for the more sophisticated tagging approaches (when compared to training on the entire Treebank).

In addition to training more sophisticated taggers on larger, tailored subsets of the Penn Treebank corpora, we should revisit our initial corpus construction decisions to see how they impact tagger performance. For example, we tagged emoticons and chat abbreviations as interjections. However, their distribution in chat is probably different than that of traditional interjections in spoken or written language. The use of one or two new tags to represent emoticons and chat abbreviations may provide a critical distinction between those and the traditional interjections that also occur in chat, e.g., greetings, yes/no responses to questions, fillers, etc. Recognizing these distinctions with a new tag(s) could improve overall performance.

Another of our early decisions that should be reconsidered is the lack of tokenization of contractions. Recall that based on their frequency of use, we treated words like "doncha" as a single word, and assigned it a single part-of-speech tag that most closely resembled its use. Thus, the post "doncha feel good?" would be tagged as "doncha/VBP feel/VB good/JJ ?/." That tag sequence would be the same as "do/VBP feel/VB good/JJ ?/.", and yet this is unlikely to be found in even the most informal written or transcribed spoken domains. Tokenizing "doncha" as "do" and "ncha" would lead to the following tagging sequence: "do/VBP ncha/PRP feel/VB good/JJ ?/.", which is a tag sequence much more likely to be found in the written and transcribed spoken domains. Both HMM and Brill taggers should be able to take advantage of this closer match to those domains. Of course, this would complicate the tokenizing task, requiring a dictionary of these contractions so that they can be recognized and split appropriately during the tokenization phase.

Finally, we should reconsider how we handle misspellings, both from a corpus construction as well as a part-of-speech tagging system approach. Including misspelled tokens in the corpus add additional labels to types, thus increasing the part-of-speech ambiguity for word types such as "there" and "your", which are both correct and incorrect spellings depending on their context. During corpus construction, these

misspellings could be corrected, but the part-of-speech tagger will certainly not actually be used in such a pristine environment. A spelling module, which both detects and attempts to correct misspelled tokens, could serve as an input to the part-of-speech tagging system. Of course, such a module would also need to be trained, with more sophisticated approaches potentially requiring part-of-speech labels as input! Thus, automated spelling correctors would complicate the real-time use of natural language processing applications that rely on part-of-speech tagging. Jurafsky and Martin provide a nice overview of misspelling recognition and correction techniques [14].

With the presentation of our experiment results for part-of-speech tagging complete, we now turn to the results of our chat dialog act classification experiments.

C. CHAT DIALOG ACT CLASSIFICATION RESULTS

Before presenting the chat dialog act classification results of the two machine-learning approaches, we first present the dialog act class counts for the chat privacy-masked corpus as well as the comparison methodology we used to assess whether the difference in machine-learning approaches is significant.

Class	Count	Percent
Statement	3163	29.93%
System	2630	24.89%
Greet	1438	13.61%
Emotion	1046	9.90%
Wh-Question	538	5.09%
Yes/No Question	538	5.09%
Accept	238	2.25%
Bye	195	1.85%
Emphasis	189	1.79%
Continuer	171	1.62%
Reject	160	1.51%
Yes Answer	109	1.03%
No Answer	73	0.69%
Other	41	0.39%
Clarify	38	0.36%
All Classes	10567	100.00%

Table 21. Chat Dialog Act Frequencies

As can be seen in Table 21, *Statement* is the most common class, followed closely by *System*, and then dropping off quickly to *Greet*, *Emotion*, *Wh-* and *Yes/No Question* classes. The remaining nine classes all occur with less than 2.25% frequency. That means only 11.5% of the posts represent 60% of th e chat dialog act class categories. This may present a problem for the machine-learning approaches, since both back propagation neural networks and the Naïve Bayes classifier require training data to make their classifications, and relatively little data is available for these categories. That being said, if there are good features that clearly distinguish these categories from higher percentage ones, there is the opportunity for the machine-learning method to make the correct classification.

As mentioned at the end of Chapter III, we divided the privacy-masked chat corpus into 30 different training/test configurations, randomly selecting 10% of the corpus (1,060 posts) to serve as the test set, and the remaining 90% to serve as the

training set. After testing each test set with the specific machine-learning approach, we calculated precision, recall, and f-scores for each dialog act class as well as the overall accuracy. A useful way to visualize the performance of the learning approach is through a confusion matrix. A confusion matrix is an $N \times N$ matrix, where N is the number of categories a test instance can be classified into. Thus, for chat dialog acts, $N = 15$. The sums of each row represent the truth, i.e., the actual counts of the classes in the test set. The sums of each column represent what the learning algorithm labeled as that class. Thus, entries on the diagonal are the number of instances labeled correctly, and recall/precision for each class can be calculated by dividing the diagonal entry by the row/column sum, respectively. Although we will not present all confusion matrices for all training/test sets, an example of one is shown in Figure 11.

	Accept	Bye	Clarify	Continuer	Emotion	Emphasis	Greet	nAnswer	Other	Reject	Statement	System	whQuestion	yAnswer	ynQuestion	Total Actua	Precision	Recall	F-score
Accept	13	0	0	0	2	1	0	0	0	0	6	0	0	0	0	22	0.722	0.591	0.65
Bye	0	12	0	0	0	1	0	0	0	0	2	0	1	0	0	16	0.923	0.75	0.828
Clarify	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	undef	0	undef
Continuer	0	0	0	0	0	2	0	0	0	0	14	0	0	0	0	16	undef	0	undef
Emotion	0	0	0	0	110	2	0	0	0	0	0	0	0	0	0	112	0.827	0.982	0.898
Emphasis	1	0	0	0	1	9	0	0	0	0	5	1	0	0	0	17	0.409	0.529	0.462
Greet	0	0	0	0	12	5	114	0	0	0	10	0	2	0	3	146	0.934	0.781	0.851
nAnswer	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	8	undef	0	undef
Other	0	0	0	0	0	0	0	0	4	0	1	0	0	0	0	5	0.8	0.8	0.8
Reject	0	0	0	0	0	0	1	0	0	0	15	0	0	0	0	16	undef	0	undef
Statement	2	1	0	0	8	2	6	0	0	0	292	2	5	0	3	321	0.785	0.91	0.843
System	0	0	0	0	0	0	0	0	0	0	4	252	0	0	0	256	0.988	0.984	0.986
whQuestion	0	0	0	0	0	0	0	0	0	0	5	0	41	0	6	52	0.804	0.788	0.796
yAnswer	2	0	0	0	0	0	0	0	1	0	0	0	0	0	1	4	undef	0	undef
ynQuestion	0	0	0	0	0	0	1	0	0	0	9	0	2	0	53	65	0.803	0.815	0.809
Total Labeled	18	13	0	0	133	22	122	0	5	0	372	255	51	0	66				
																Test Set Accuracy:	0.851		

Figure 11. Example Confusion Matrix for Chat Dialog Act Classification (Back Propagation Neural Network, 24 Features, 100 iterations)

Finally, we calculated the means and standard deviations for the recall, precision, f-score, and overall accuracy of each experiment configuration. To ascertain if there was a significant difference in the performance of two learning approaches, we performed hypothesis (z) tests using the approaches mean and standard deviations. For 95% confidence, we reject the null hypothesis that the means are equal if $|z| > 1.96$.

With our discussion of chat dialog act class frequencies and comparison methodology complete, we now turn to the classification results. We will first look at the

performance of the back propagation neural network and Naïve Bayes classifier using 27 features. We will then look at the performance of both learning approaches using a smaller, 24 feature set.

1. 27 Feature Experiment Results

For the 27 feature set, we first present the results of the back propagation neural network, to include the effect of varying the number of training iterations. We then present the results of the Naïve Bayes classifier, to include the effect of ignoring the prior probability for each class in the Naïve Bayes argmax equation.

a. Back Propagation Neural Network

The mean and standard deviation of each chat dialog act class's precision, recall, and f-scores as well as the overall accuracy for the back propagation neural network trained for 100 iterations are shown below in Table 22.

	Precision		Recall		F-Score	
	Mean	**St Dev**	**Mean**	**St Dev**	**Mean**	**St Dev**
Accept	undef	undef	0.237	0.111	undef	undef
Bye	0.803	0.096	0.785	0.093	0.788	0.064
Clarify	undef	undef	undef	undef	undef	undef
Continuer	undef	undef	0.008	0.024	undef	undef
Emotion	0.775	0.041	0.955	0.041	0.855	0.034
Emphasis	0.641	0.108	0.613	0.1	0.619	0.075
Greet	0.921	0.036	0.83	0.039	0.873	0.031
No Answer	undef	undef	0.014	0.078	undef	undef
Other	0.891	0.174	0.86	0.169	0.857	0.136
Reject	undef	undef	0.059	0.1	undef	undef
Statement	0.75	0.028	0.861	0.022	0.801	0.016
System	0.985	0.008	0.983	0.008	0.984	0.005
Wh-Question	0.809	0.047	0.796	0.059	0.801	0.042
Yes Answer	undef	undef	0	0	undef	undef
Yes/No Question	0.747	0.049	0.801	0.036	0.772	0.03
Overall Accuracy	0.828	0.012	-	-	-	-

Table 22. Back Propagation Neural Network Classifier Performance (27 Features, 100 iterations)

The overall accuracy of 82.8% is a significant improvement over both choosing randomly (6.7% given 15 choices) and choosing the MLE (29.9% for *Statement*). Classes performing particular well include *System* (f-score of 0.984) and *Emotion* (f-score of 0.855). This is not surprising, since both classes have very strong features associated with them. Overall, the six most frequent classes, representing nearly 90% of the posts, performed well, with average f-scores (over the 30 training/test sets) of 0.772 or greater. We were also able to detect the lower frequency *Other* (f-score of 0.857) and *Emphasis* (f-score of 0.619) classes.

However, for all other lower frequency classes we were unable to reliably assign a classification. This is somewhat disappointing, because we believe we had good features to detect *Yes-/No- Answers* (features f2, f15, and f16 from Table 11),

Accepts/Rejects (features f15 and f16), *Continuers* (feature f11), and *Clarifies* (feature f1 and f25). It appears that the back propagation neural network mislabeled most of the lower frequency class posts as *Statements*. This evidenced by the mean precision score of 0.75 for the *Statement* class. This indicates that on average, one quarter of those posts labeled as *Statements* were not. A specific example of this can be seen in one of the confusion matrices we generated in the 27 feature back propagation neural network, shown in Figure 12. Notice the large number of low frequency posts mislabeled as *Statements*, as indicated in the *Statement* column.

	Accept	Bye	Clarify	Continuer	Emotion	Emphasis	Greet	nAnswer	Other	Reject	Statement	System	whQuestion	yAnswer	ynQuestion	Total Actua	Precision	Recall	F-score
Accept	4	2	0	0	1	0	0	0	0	0	8	1	0	0	0	16	0.4	0.25	0.308
Bye	0	17	0	0	0	0	0	0	0	0	6	0	0	0	0	23	0.85	0.739	0.791
Clarify	0	0	0	0	1	0	0	0	0	0	4	0	0	0	0	5	undef	0	undef
Continuer	0	0	0	0	0	0	1	0	0	0	14	0	0	0	3	18	undef	0	undef
Emotion	0	0	0	0	101	0	0	0	0	0	4	0	0	0	0	105	0.727	0.962	0.828
Emphasis	0	0	0	0	1	8	0	0	0	0	5	0	0	0	0	14	0.727	0.571	0.64
Greet	0	0	0	0	10	0	116	0	0	0	17	1	2	0	0	146	0.928	0.795	0.856
nAnswer	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	3	undef	0	undef
Other	0	0	0	0	0	0	0	0	4	0	2	0	0	0	0	6	1	0.667	0.8
Reject	0	0	0	0	2	0	0	0	0	0	6	1	0	0	0	9	0	0	undef
Statement	1	1	0	0	23	1	8	0	0	0	281	2	3	0	8	328	0.751	0.857	0.801
System	0	0	0	0	0	2	0	0	0	0	7	256	0	0	0	265	0.981	0.966	0.973
whQuestion	0	0	0	0	0	0	0	0	0	0	7	0	43	0	5	55	0.896	0.782	0.835
yAnswer	5	0	0	0	0	0	0	0	0	0	4	0	0	0	0	9	undef	0	undef
ynQuestion	0	0	0	0	0	0	0	0	0	0	8	0	0	0	47	55	0.746	0.855	0.797
Total Labeled	10	20	0	0	139	11	125	0	4	2	374	261	48	0	63				
																Test Set Accuracy:		0.83	

Figure 12. Example Confusion Matrix for Chat Dialog Act Classification (Back Propagation Neural Network, 27 Features, 100 iterations)

Our first attempt to improve the performance of the back propagation neural network with 27 features was to increase the number of training iterations for each training/test set. The longer the neural network is allowed to train, the more the overall error is reduced between the target output values and the output unit layer. However, neural networks are susceptible to overtraining, such that they will continue to reduce the training set error at the expense of the domain in general, as represented by the test set. To ascertain when overtraining begins to occur, we ran a sample test on a smaller training/test set (3,507 posts total) using only 22 features. The errors on the training/test set as a function of the number of iterations are shown in Figures 13 and 14.

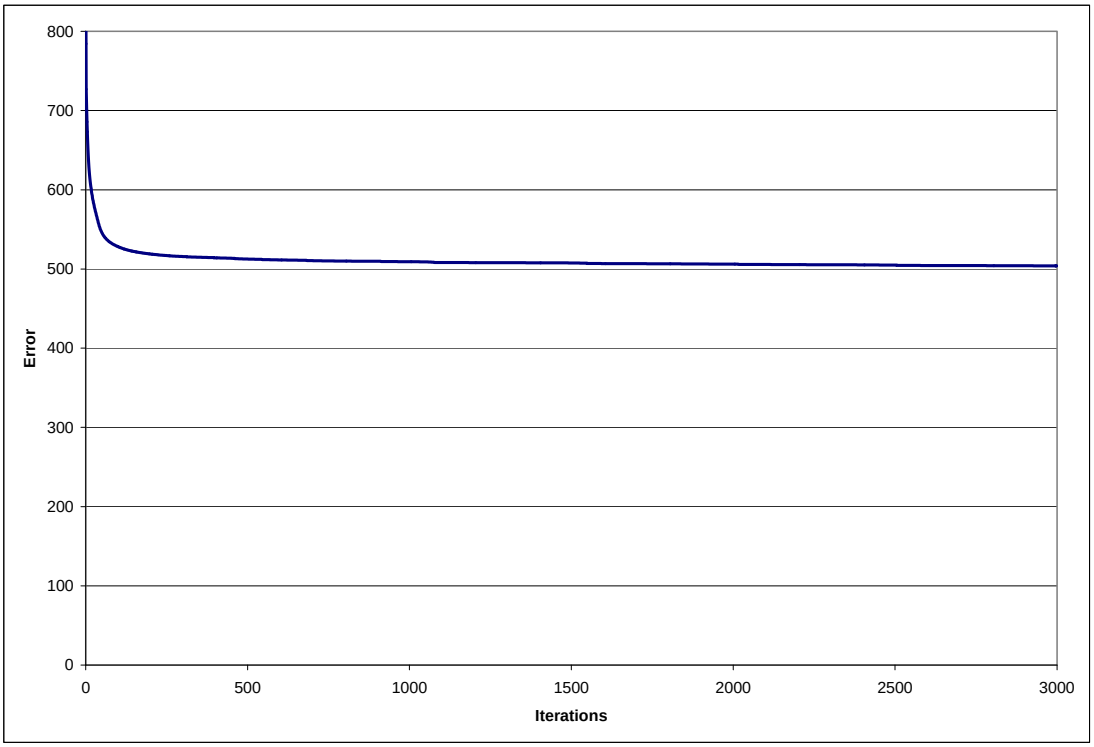

Figure 13. Back Propagation Neural Network Training Set Error (3007 posts, 22 Features)

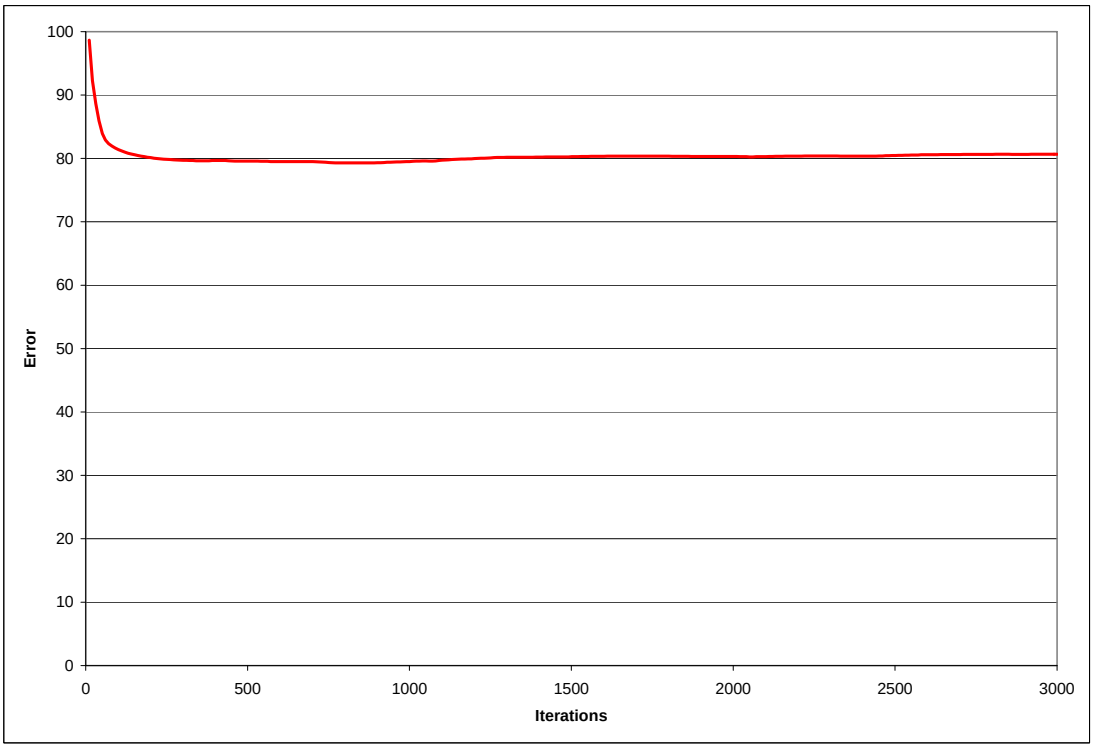

Figure 14. Back Propagation Neural Network Test Set Error (500 posts, 22 Features)

As can be seen, most of the error on the test set is reduced by roughly 250 iterations. Also, although the training data error rate continues to decrease as iterations increase, at roughly 850 iterations, the test data error starts to increase. Although this is not a formal assessment of when over-fitting begins to occur, it suggests that a large number of training iterations are not required for back propagation neural networks of this size to reach maximum expected error reduction.

With this in mind, we ran an excursion on our back propagation neural network with 27 features, training for 300 iterations. We present the mean and standard deviation performance (as represented by class f-scores and overall accuracy) of the 100 and 300 iteration versions in Table 23.

	BPNN F-Score: 100 Iterations		**BPNN F-Score: 300 Iterations**		
	Mean	**Std Dev**	**Mean**	**Std Dev**	**|z|**
Accept	undef	undef	undef	undef	undef
Bye	0.788	0.064	0.788	0.066	0.002
Clarify	undef	undef	undef	undef	undef
Continuer	undef	undef	undef	undef	undef
Emotion	0.855	0.034	0.863	0.033	0.947
Emphasis	0.619	0.075	0.631	0.069	0.682
Greet	0.873	0.031	0.874	0.030	0.178
No Answer	undef	undef	undef	undef	undef
Other	0.857	0.136	0.857	0.136	0.000
Reject	undef	undef	undef	undef	undef
Statement	0.801	0.016	0.806	0.016	1.139
System	0.984	0.005	0.984	0.005	0.508
Wh-Question	0.801	0.042	0.804	0.042	0.235
Yes Answer	undef	undef	undef	undef	undef
Yes/No Question	0.772	0.030	0.770	0.033	0.165
Overall Accuracy	0.828	0.012	0.831	0.012	0.981

Table 23. Back Propagation Neural Network Classifier F-Score Comparison (27 Features, 100 vs. 300 iterations)

In 14 of the 15 categories, mean performance either stayed the same or improved via training three times longer. Mean overall accuracy also improved by

training longer. However, none of the performance measures improved to a degree that we can confidently state that training for 300 iterations provides better results than training for only 100. In addition, at a summary level, there was no indication that we are picking up lower frequency classes any better. Thus, more work needs to be done to improve this aspect of performance.

With our initial discussion on the performance of the back propagation neural network complete, we now turn to the Naïve Bayes classifier experimental results.

b. Naïve Bayes Classifier

The mean and standard deviation of each chat dialog act class's precision, recall, and f-scores as well as the overall accuracy for the Naïve Bayes classifier are shown below in Table 24.

	Precision		Recall		F-Score	
	Mean	**St Dev**	**Mean**	**St Dev**	**Mean**	**St Dev**
Accept	0.266	0.16	0.074	0.048	undef	undef
Bye	0.82	0.116	0.56	0.1	0.658	0.081
Clarify	undef	undef	undef	undef	undef	undef
Continuer	0.394	0.229	0.115	0.072	undef	undef
Emotion	0.838	0.035	0.765	0.043	0.799	0.03
Emphasis	0.631	0.226	0.216	0.077	0.314	0.104
Greet	0.824	0.036	0.852	0.032	0.837	0.028
No Answer	undef	undef	0.061	0.105	undef	undef
Other	undef	undef	0.33	0.294	undef	undef
Reject	undef	undef	0.062	0.066	undef	undef
Statement	0.634	0.024	0.857	0.019	0.729	0.018
System	0.951	0.012	0.952	0.015	0.951	0.01
Wh-Question	0.738	0.061	0.577	0.067	0.645	0.056
Yes Answer	undef	undef	0.091	0.098	undef	undef
Yes/No Question	0.72	0.065	0.477	0.069	0.571	0.061
Overall Accuracy	0.761	0.013	-	-	-	-

Table 24. Naïve Bayes Classifier Performance (27 Features)

As with the back propagation neural network, the overall accuracy of 76.1% is a significant improvement over both choosing randomly (6.7% given 15 choices) and choosing the MLE (29.9% for *Statement*). Classes performing particular well include *System* (f-score of 0.951) and *Greet* (f-score of 0.837). However, only three of the six most frequent classes had f-scores above 0.799. Overall, the Naïve Bayes classifier performed significantly worse than the back propagation neural network trained on the same features, And, as with the back propagation neural network, the Naïve Bayes classifier was unable to reliably assign a classification to lower frequency classes.

Of note, the Naive Bayes classifier also mislabeled several classes as *Statement* as is evident by its precision value of 0.634. The confusion matrix depicted in Figure 15, representative of the Naïve Bayes classifier, highlights this fact. Again, note the *Statement* column values.

	Accept	Bye	Clarify	Continuer	Emotion	Emphasis	Greet	nAnswer	Other	Reject	Statement	System	whQuestior	yAnswer	ynQuestion	Total Actua	Precision	Recall	F-score
Accept	1	0	0	0	1	0	1	0	0	0	13	0	0	0	0	16	0.2	0.063	0.095
Bye	0	11	0	0	0	0	1	0	0	0	11	0	0	0	0	23	1	0.478	0.647
Clarify	0	0	0	0	1	0	0	0	0	0	4	0	0	0	0	5	undef	0	undef
Continuer	0	0	0	2	0	0	0	0	0	0	13	1	2	0	0	18	0.667	0.111	0.19
Emotion	1	0	0	0	74	0	3	0	0	0	26	1	0	0	0	105	0.881	0.705	0.783
Emphasis	0	0	0	0	2	5	1	0	0	0	6	0	0	0	0	14	0.625	0.357	0.455
Greet	0	0	0	0	1	0	124	0	0	0	18	1	2	0	0	146	0.838	0.849	0.844
nAnswer	0	0	0	0	0	0	0	1	0	0	2	0	0	0	0	3	0.5	0.333	0.4
Other	0	0	0	0	0	0	0	0	2	0	2	2	0	0	0	6	1	0.333	0.5
Reject	0	0	0	0	0	0	0	1	0	0	8	0	0	0	0	9	undef	0	undef
Statement	1	0	0	1	5	0	16	0	0	0	289	3	5	0	8	328	0.652	0.881	0.75
System	0	0	0	0	0	2	0	0	0	0	11	251	0	0	1	265	0.969	0.947	0.958
whQuestion	0	0	0	0	0	0	2	0	0	0	17	0	32	0	4	55	0.681	0.582	0.627
yAnswer	2	0	0	0	0	1	0	0	0	0	6	0	0	0	0	9	undef	0	undef
ynQuestion	0	0	0	0	0	0	0	0	0	0	17	0	6	0	32	55	0.711	0.582	0.64
Total Labeled	5	11	0	3	84	8	148	2	2	0	443	259	47	0	45				
																Test Set Accuracy:	0.78		

Figure 15. Example Confusion Matrix for Chat Dialog Act Classification (Naïve Bayes, 27 Features)

Although the Naïve Bayes classifier is mislabeling many classes as the MLE (*Statement*), there is an explicit way to remove this effect. Specifically, the prior probability term for each class, $P(C_i)$ can be removed from the Naïve Bayes classifier equation, leaving

$$\hat{C}_{\text{No Prior}} = \arg\max_{C_i \in Classes} \prod_j P\left(f_j \mid C_i\right)$$

To ascertain the effect of this, we ran an excursion on our Naïve Bayes classifier, this time removing the effect of the prior class probability. The mean and standard deviation performance measures for both versions of Naïve Bayes classifier are presented in the Table 25.

	Naïve Bayes F-Score:		**Naïve Bayes F-Score: No Prior**		
	Mean	**Std Dev**	**Mean**	**Std Dev**	**|z|**
Accept	undef	undef	undef	undef	undef
Bye	0.658	0.081	0.607	0.069	2.654
Clarify	undef	undef	undef	undef	undef
Continuer	undef	undef	0.266	0.075	undef
Emotion	0.799	0.030	0.820	0.026	2.918
Emphasis	0.314	0.104	0.443	0.095	5.012
Greet	0.837	0.028	0.823	0.028	1.978
No Answer	undef	undef	undef	undef	undef
Other	undef	undef	undef	undef	undef
Reject	undef	undef	undef	undef	undef
Statement	0.729	0.018	0.669	0.025	10.657
System	0.951	0.010	0.954	0.009	1.034
Wh-Question	0.645	0.056	0.676	0.049	2.302
Yes Answer	undef	undef	undef	undef	undef
Yes/No Question	0.571	0.061	0.630	0.046	4.242
Overall Accuracy	0.761	0.013	0.729	0.013	9.702

Table 25. Naïve Bayes Classifier F-Score Comparison (27 Features, Prior Class Probability Included/Not Included)

As can be seen, there are significant differences between nearly all the classes f-scores. Performance improved in the *Continuer*, *Emotion*, *Emphasis*, *Wh-Question* and *Yes/No Question* classes by removing the prior. However, performance was degraded in the *Bye*, *Greet*, and *Statement* classes. In particular, *Statement*'s f-score dropped from 0.729 to 0.669 by removing the prior probability term. Since this was the largest class, it had the overall effect of offsetting the f-score improvements in the other classes, significantly reducing overall accuracy from 76.1% to 72.9%.

The actual effect of removing the prior can be visualized by looking at the change in the confusion matrix in Figure 16, using the same training and test sets as presented in Figure 15.

	Accept	Bye	Clarify	Continuer	Emotion	Emphasis	Greet	nAnswer	Other	Reject	Statement	System	whQuestior	yAnswer	ynQuestion	Total Actua	Precision	Recall	F-score
Accept	4	0	0	0	1	0	0	0	0	0	9	0	0	1	1	16	0.2	0.25	0.222
Bye	1	14	0	0	0	1	0	0	0	1	6	0	0	0	0	23	0.609	0.609	0.609
Clarify	0	0	2	1	0	0	0	0	0	0	2	0	0	0	0	5	0.667	0.4	0.5
Continuer	1	0	0	4	0	0	0	0	0	0	10	0	3	0	0	18	0.167	0.222	0.19
Emotion	2	1	0	0	85	0	4	0	1	0	11	1	0	0	0	105	0.833	0.81	0.821
Emphasis	0	0	0	1	1	8	1	0	0	0	3	0	0	0	0	14	0.5	0.571	0.533
Greet	1	1	1	1	5	1	125	0	1	0	7	0	2	1	0	146	0.772	0.856	0.812
nAnswer	0	0	0	0	0	0	0	2	0	0	1	0	0	0	0	3	0.333	0.667	0.444
Other	0	0	0	0	0	1	0	0	3	0	0	2	0	0	0	6	0.5	0.5	0.5
Reject	0	1	0	1	1	0	0	1	0	1	4	0	0	0	0	9	0.1	0.111	0.105
Statement	7	5	0	12	9	1	31	3	1	7	215	3	9	4	21	328	0.736	0.655	0.694
System	1	0	0	1	0	3	0	0	0	1	8	250	0	0	1	265	0.977	0.943	0.96
whQuestion	0	0	0	2	0	0	1	0	0	0	8	0	39	0	5	55	0.65	0.709	0.678
yAnswer	3	0	0	0	0	1	0	0	0	0	5	0	0	0	0	9	0	0	undef
ynQuestion	0	1	0	1	0	0	0	0	0	0	3	0	7	0	43	55	0.606	0.782	0.683
Total Labeled	20	23	3	24	102	16	162	6	6	10	292	256	60	6	71				
																Test Set Accuracy:	0.752		

Figure 16. Example Confusion Matrix for Chat Dialog Act Classification (Naïve Bayes, 27 Features, No Prior Probability Term)

Removing the prior probability term permits other classes to be recognized, yet significantly reduces the recall of the *Statement* class (from 289 actual *Statements* labeled as such to 215 in this example).

With our initial discussion of the machine-learning approaches complete, we now turn to the effect of reducing the number of features for the methods to consider.

2. 24 Feature Experiment Results

As we noted earlier, some features that were intended to pick up the lower frequency classes did not appear to work. Before modifying the feature set to pick up these classes, we first removed those ineffective features to see how it impacted the overall performance of the learning approaches. Specifically, we removed feature f1 (number of posts ago the poster made a spelling error), f25 (a word is an "even" or "mean" variant), and f26 (total number of users currently in the chat room).

We first present the results of the back propagation neural network for this smaller feature set. We then present the results of the Naïve Bayes classifier using the smaller feature set.

a. Back Propagation Neural Network

The precision, recall, and f-scores for the 24 feature version of the back propagation neural network trained for 300 iterations are presented in Table 26. A comparison with 27 feature network trained for 300 iterations is presented in Table 27.

	Precision		Recall		F-Score	
	Mean	**St Dev**	**Mean**	**St Dev**	**Mean**	**St Dev**
Accept	undef	undef	0.289	0.153	undef	undef
Bye	0.815	0.098	0.816	0.089	0.812	0.075
Clarify	undef	undef	undef	undef	undef	undef
Continuer	undef	undef	0.013	0.034	undef	undef
Emotion	0.789	0.043	0.95	0.023	0.862	0.028
Emphasis	0.648	0.134	0.66	0.147	0.635	0.108
Greet	0.936	0.023	0.833	0.037	0.881	0.023
No Answer	undef	undef	0	0	undef	undef
Other	0.887	0.185	0.834	0.208	0.832	0.161
Reject	undef	undef	0.068	0.107	undef	undef
Statement	0.746	0.03	0.876	0.019	0.806	0.019
System	0.985	0.014	0.985	0.007	0.985	0.006
Wh-Question	0.823	0.054	0.801	0.066	0.811	0.052
Yes Answer	undef	undef	0	0	undef	undef
Yes/No Question	0.777	0.05	0.809	0.046	0.791	0.037
Overall Accuracy	0.832	0.009	-	-	-	-

Table 26. Back Propagation Neural Network Classifier Performance (24 Features, 300 iterations)

	BPNN F-Score: 300 Its, 24 Feats		BPNN F-Score: 300 Its, 27 Feats		
	Mean	**Std Dev**	**Mean**	**Std Dev**	**|z|**
Accept	undef	undef	undef	undef	undef
Bye	0.812	0.075	0.788	0.066	1.291
Clarify	undef	undef	undef	undef	undef
Continuer	undef	undef	undef	undef	undef
Emotion	0.862	0.028	0.863	0.033	0.211
Emphasis	0.635	0.108	0.631	0.069	0.152
Greet	0.881	0.023	0.874	0.030	0.929
No Answer	undef	undef	undef	undef	undef
Other	0.832	0.161	0.857	0.136	0.675
Reject	undef	undef	undef	undef	undef
Statement	0.806	0.019	0.806	0.016	0.022
System	0.985	0.006	0.984	0.005	0.231
Wh-Question	0.811	0.052	0.804	0.042	0.550
Yes Answer	undef	undef	undef	undef	undef
Yes/No Question	0.791	0.037	0.770	0.033	2.364
Overall Accuracy	0.832	0.009	0.831	0.012	0.482

Table 27. Back Propagation Neural Network Classifier F-Score Comparison (24 Features vs. 27 Features, 300 Iterations)

As can be seen in the comparison table, for the most part there were no significant changes in any of the f-scores. However, there was a significant improvement in *Yes/No Question* classification (f-score of 0.791), an important chat dialog act category. Moreover, the overall performance of the 24 feature, 300 iteration back propagation neural network (83.2% accuracy) is significantly better than its 27 feature, 100 iteration counterpart (82.8% accuracy).

Thus, the removal of three features to the back propagation neural network appears to have no impact on overall performance. This is an important finding, since we have identified features that appear to be unnecessary in the classification decision process. We now turn to the impact of removing those three features on the Naïve Bayes classifier.

b. Naïve Bayes Classifier

The precision, recall, and f-scores for the 24 feature version of the Naïve Bayes classifier are presented in Table 28. A comparison with 27 feature version is presented in Table 29.

	Precision		Recall		F-Score	
	Mean	**St Dev**	**Mean**	**St Dev**	**Mean**	**St Dev**
Accept	0.346	0.198	0.104	0.067	undef	undef
Bye	0.864	0.084	0.573	0.12	0.681	0.1
Clarify	undef	undef	undef	undef	undef	undef
Continuer	0.345	0.291	0.084	0.071	undef	undef
Emotion	0.827	0.045	0.833	0.038	0.829	0.033
Emphasis	0.558	0.216	0.244	0.105	0.33	0.125
Greet	0.848	0.027	0.847	0.036	0.847	0.027
No Answer	undef	undef	0.099	0.098	undef	undef
Other	undef	undef	0.276	0.289	undef	undef
Reject	0.344	0.357	0.058	0.058	undef	undef
Statement	0.638	0.025	0.869	0.021	0.736	0.02
System	0.967	0.012	0.951	0.013	0.959	0.008
Wh-Question	0.735	0.074	0.618	0.08	0.668	0.061
Yes Answer	undef	undef	0.089	0.092	undef	undef
Yes/No Question	0.762	0.074	0.526	0.068	0.62	0.06
Overall Accuracy	0.773	0.014	-	-	-	-

Table 28. Naïve Bayes Classifier Performance (24 Features)

	Naïve Bayes F-Score: 24 Feats		Naïve Bayes F-Score: 27 Feats		
	Mean	**Std Dev**	**Mean**	**Std Dev**	**|z|**
Accept	undef	undef	undef	undef	undef
Bye	0.681	0.100	0.658	0.081	0.982
Clarify	undef	undef	undef	undef	undef
Continuer	undef	undef	undef	undef	undef
Emotion	0.829	0.033	0.799	0.030	3.714
Emphasis	0.330	0.125	0.314	0.104	0.542
Greet	0.847	0.027	0.837	0.028	1.350
No Answer	undef	undef	undef	undef	undef
Other	undef	undef	undef	undef	undef
Reject	undef	undef	undef	undef	undef
Statement	0.736	0.020	0.729	0.018	1.343
System	0.959	0.008	0.951	0.010	3.262
Wh-Question	0.668	0.061	0.645	0.056	1.522
Yes Answer	undef	undef	undef	undef	undef
Yes/No Question	0.620	0.060	0.571	0.061	3.116
Overall Accuracy	0.773	0.014	0.761	0.013	3.352

Table 29. Naïve Bayes Classifier F-Score Comparison (24 Features vs. 27 Features)

As can be seen in the comparison table, there are f-score improvements in the 24 feature Naïve Bayes classifier across the board. In particular, there are significant improvements in *Emotion* (f-score of 0.829), *System* (f-score of 0.959), and *Yes/No Question* (f-score of 0.620) categories. These improvements led to a significant improvement in the Naïve Bayes classifier's overall accuracy (77.3%, up from 76.1% for the 27 feature version).

With our presentation of the chat dialog act classification results complete, we now turn to a general discussion of the leaning task as well as potential improvements to the classifier learning approaches.

3. Discussion

Unfortunately, time did not permit us to formally examine the misclassified posts. However, we noticed that for both learning methods, several of the "second-highest" classification scores on the test set were in fact the "true" dialog act class label. In addition, we noticed that some of the incorrect classification decisions that both learning approaches made were arguably "correct". By this we mean that a different human annotator could easily arrive at the same conclusion that the machine-learning approach reached. Thus, the chat dialog act experiment results, along with these informal findings, lead to several avenues for improvement.

First, we could relax the condition that a post can only hold one chat dialog act class label. Obviously, the original simplifying assumption of one dialog act class per post is not a perfect fit for what actually occurs. For example, by its very nature a single post can potentially contain a greeting to one person, followed by asking a question to another, followed by rejecting a statement of a third person. Thus, permitting a post to have multiple dialog act labels addresses this issue.

A better approach, however, would be to segment chat dialog acts at a finer level. Segmenting at this "utterance" level would provide a less ambiguous decision for the classifier to make, perhaps improving its performance. However, based on the split turn phenomenon characterized by Zitzen and Stein, utterances are not necessarily limited to the confines of a single post, and may in fact span two or more posts [5]. Thus, while segmenting at the utterance level may improve the classifier's performance when considered alone, overall system performance may suffer due to the more difficult utterance segmentation phase. There are methods to segment at the utterance level; as discussed in Chapter II, Ivanovic developed an approach for dialog act classification of instant messaging (IM) systems [26]. That being said, his task was somewhat easier, since in IM there are only two participants with one thread of conversation going on at a time. Segmentation at the utterance level for chat might require the separation of the various conversation threads first, which is an area of active research in and of itself. Nevertheless, Ivanovic's and others' utterance segmentation approaches better match

actual discourse structure, and thus merit serious consideration for improving the performance of the chat dialog act classification approaches.

Another avenue for classification improvement includes better selection of the chat dialog classes themselves. As noted in Chapters 2 and 3, Stolcke et al defined 42 dialog act classes for spoken conversation. For Wu et al's purposes (as well as ours), many of Stolcke et al's classes were collapsed into a single chat dialog act class, *Statement*. Since *Statement* had a low precision and yet was the highest frequency class, dividing it up into more specific classes (e.g. *Opinion* as well as *Statement*) should help the classifiers in making decisions. This is because the resulting, more specific, class's prior probabilities will be lower. In addition, even though the high frequency *System* dialog act classification was quite successful, it too should be divided up. This is because it contained a number of phenomena that deserve better discrimination, e.g. commands to the chat room system/chatbots versus system/chatbot responses. Of course, additional classes require either additional or better features to help discriminate between them.

We took a supervised approach in our original selection of features to measure, e.g., we knew that *System* posts contained specific words in all-capital letters that we automatically identified during the training phase. That being said, it is worthwhile to consider unsupervised feature learning. For example, simple unigram and/or bigram frequencies alone might permit better discrimination. We in fact used this approach, albeit in a targeted fashion. For example, features that identified words found in *Greets*, *Byes*, *Emotions*, *Yes/No Answers*, and *Accepts/Rejects* (f3-f7 and f12-f16 from Table 11) were actually collected by identifying words tagged as "UH" in the training sets within those post categories. Permitting the Naïve Bayes classifier to identify and determine probabilities for all unigrams/bigrams across a training set might enable better discrimination of lower frequency chat dialog act classes in a test set.

Finally, combined with better classes and features, different machine-learning approaches may permit better classification. For example, case-based reasoning, which measures the "distances" of the instance to be classified from those in a labeled database, could provide more accurate classification of low frequency classes. That being said, the number of comparisons to make (e.g., distance to a single neighbor, k-nearest neighbors,

class mean, etc.) as well as the distance measure definition itself (e.g. Euclidean, city-block, etc) will have an impact on classification performance. A fuller description of case-based reasoning approaches can be found in Mitchell [23] and Luger [24]. Another learning method that bears consideration is the use of HMMs. Stolcke et al used HMMs to identify the most likely sequence of dialog act classes in a conversation [17]. In that case, the dialog acts were the hidden states, while features of the utterances were the observed sequence. However, Stolcke et al were dealing with Switchboard conversations in series; chat involves multiple, interleaved conversations in parallel. Thus, use of this approach may require the separation of conversation threads first.

With the presentation of our experiment results complete, we conclude with summary of our results and recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

V. SUMMARY AND FUTURE WORK

A. SUMMARY

During the course of our research, we preserved 477,835 chat posts and associated user profiles in an XML format for future investigation. We privacy-masked 10,567 of those posts, permitting other researchers to replicate and improve upon our results. We annotated each of the privacy-masked corpus's 45,068 tokens with a part-of-speech tag. Using the Penn Treebank, we improved part-of-speech tagging performance from 87.0% mean accuracy (HMM tagger using only chat data) to 90.8%. This represents a reduction in total error of over 29%. We also annotated each of the privacy-masked corpus's 10,567 posts with a chat dialog act. Using a neural network with 23 input features, we achieved 83.2% mean dialog act classification accuracy.

Although these results are notable based on the privacy-masked corpus's size, we believe there are a number of things that we can do to significantly improve on these results as well as extend the usefulness of the corpus for other NLP tasks. We now present this potential future work.

B. FUTURE WORK

Our recommendations for future work are broken into five tasks: 1) Improve part-of-speech tagging on the existing privacy-masked corpus; 2) Improve chat dialog act classification on the existing privacy-masked corpus; 3) Perform syntax analysis on the existing privacy-masked chat corpus; 4) Use information from the previous three tasks to perform semantic NLP tasks of entity identification/disambiguation, conversation thread detection/separation, and author profiling; and 5) Increase the size of the privacy-masked chat corpus.

1. Part-of-Speech Tagging Improvements

As discussed in Chapter IV, Section B.4, we recommend the following three actions to improve part-of-speech tagging. First, we recommend tokenizing all

contractions, including those that do not contain an apostrophe. We present a list of all contractions in the privacy-masked chat corpus that do not contain apostrophes along with an alternate tokenization in Appendix B. Tokenizing these contractions into separate words, especially high frequency ones such as "ill" (for "I will") should permit sophisticated part-of-speech taggers to take advantage of more likely tag sequences found in other domains.

Second, we recommend retagging many of the emoticons and chat abbreviations with one or possibly two new tags, as opposed to the interjection tag, "UH". Based on our observations of the privacy-masked chat corpus, emoticons and chat abbreviations generally have different distributions than interjections, and thus merit a new tag unique to the chat domain. We present a list of all emoticons and chat abbreviations found in the privacy-masked corpus in Appendices C and D, respectively. Any retagging of emoticons and abbreviations should be approached carefully, however. For example, chat abbreviations such as "wtf" and "brb" are often used as equivalents to "what" (tagged "WP") and "bye" (tagged "UH"), respectively. Thus, a simple find/replace will not suffice when retagging these abbreviations.

Finally, we recommend optimizing the amount of data used from other domains to support part-of-speech tagging. For example, if chat exhibits lexical properties more in common with written as opposed spoken domains, it may make sense to use more training data from the written domain itself. Our current best-performing taggers use all of the data provided in the Brown (written), Wall Street Journal (written), and Switchboard (transcribed spoken) corpora; adjusting these ratios could improve more sophisticated tagger performance.

2. Chat Dialog Act Classification Improvements

As discussed in Chapter IV, Section C.3, we recommend the following three actions to improve chat dialog act classification. First, we recommend the use of additional and/or better classes for the dialog acts themselves. Statements of fact and opinions are currently grouped into a single *Statement* class; we believe it makes sense to differentiate between the two by labeling each a separate class. Similarly, we believe it

makes sense to divide the *System* class up into user commands to the chat room system (and/or chatbots) as well as responses from the system (and/or chatbots). Finally, the need for a *Continuer* class is not necessary if the next action, utterance-level segmentation, is implemented.

Segmentation at the utterance-level (instead of post-level) would permit a more specific dialog act classification to be made, thus reducing the likelihood that more than two classes might apply to a single utterance. Utterance-level segmentation has been performed in both spoken and CMC domains, for example, [17] and [26], respectively. An additional benefit of utterance-level segmentation is that it might also improve part-of-speech tagging performance when training data includes non-chat domains. This is because the context associated with part-of-speech tagging should not cross sentence boundaries. And yet, a chat post can include multiple sentences. Under utterance-level segmentation, each individual sentence in a post would be a separate utterance, and could thus take better advantage of training data from non-chat domains such as the various Penn Treebank corpora.

As discussed in Chapter IV Section C, we found that some of the dialog act features we used were ineffective, and that overall accuracy actually improved once we removed them. As such, we recommend a complete review of the features used to support chat dialog act classification. In particular, we believe that there is great potential for n-gram distributions, used in conjunction with the Naïve Bayes classifier, to significantly increase classification accuracy.

3. Syntax Analysis

Throughout this research we have referred to the syntax, or structure, of language in general and chat in particular. The ability to automatically parse a sentence (or post/utterance) into a tree structure is an important step in determining its meaning. An example parse of a Wall Street Journal sentence is shown in Figure 17. Natural language syntax can be approximated by probabilistic context free grammars (PCFGs), which are simply context free grammars with probabilities attached to the production rules. As with stochastic part-of-speech taggers, these probabilities are learned during a training phase

with labeled corpora. In fact, the Penn Treebank gets its name because (in addition to part-of-speech tags) it contains parses, or trees, for each of the sentences from its various corpora. A description of how PCFGs can be applied to parsing can be found in [13] and [14].

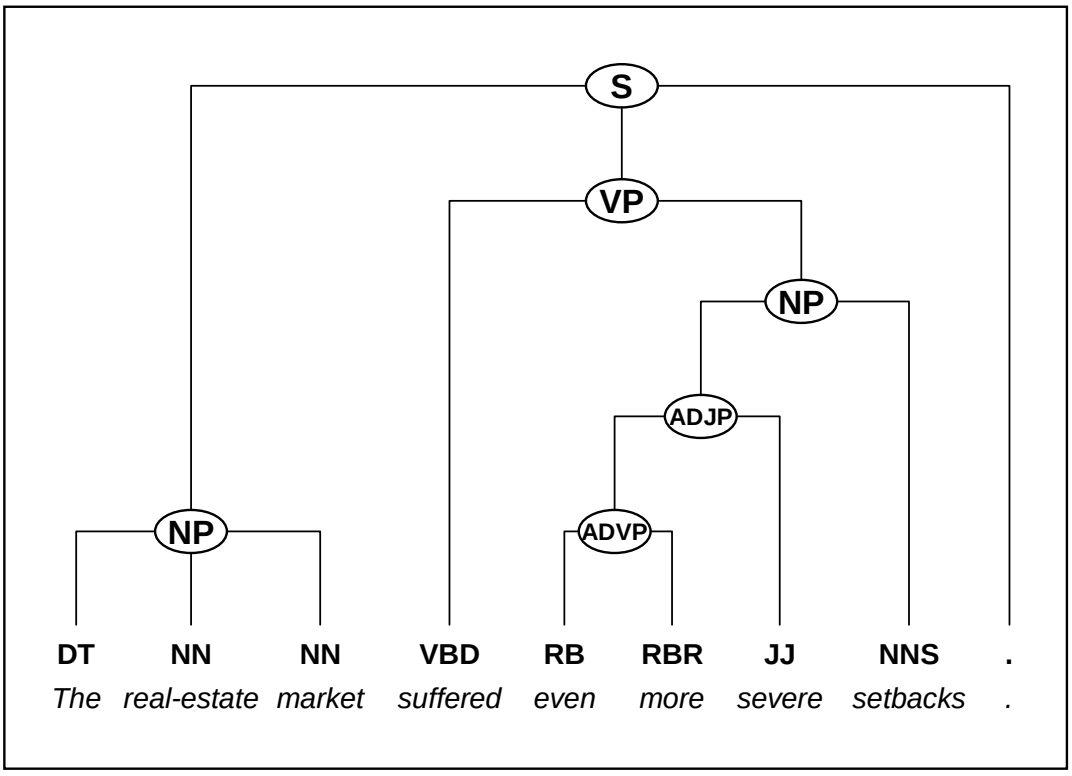

Figure 17. Example Wall Street Journal Sentence Parse (From [20])

Because of its importance to other NLP tasks, we highly recommend the addition of parses at the post and/or utterance level for the privacy-masked chat corpus. Using the same bootstrapping approach discussed in Chapter III, Section A.5, an initial parser could be trained on data from the Penn Treebank. This parser would then be used to assign initial parses to a subset of the privacy-masked corpus. These parses would then be hand-verified. Finally, a new parser would be built, trained on both Penn Treebank and chat data to bootstrap the parsing to the full privacy-masked corpus data set. Once the full data set had been parsed, a parser would then be built to optimize performance on chat based on data from chat as well as non-chat domains. Indeed, Hwa demonstrated in [27] that grammars from sparsely labeled training data (e.g., only higher-level constituent

labels for chat data) can use an adaptation strategy which produces grammars that parse almost as well as grammars induced from fully-labeled corpora.

4. Other Semantic NLP Applications

There are several other NLP tasks that can be investigated immediately with the current version of the privacy-masked chat corpus. For example, the corpus's part-of-speech and dialog act classification information can be used in conjunction with other features to improve upon Lin's author profiling work [11].

Also, there is the great potential to investigate entity disambiguation algorithms using the privacy-masked corpus as well as the corresponding original sessions that contain actual user names. As noted in Chapter III Section A.2, users are referred to both with their screen names as well as many variants of those names. This is perhaps another unique phenomenon that separates chat from both written and spoken domains. These experiments could be initiated fairly quickly, since the already-accomplished privacy-masking activity covers most of the hand-annotation effort required for entity disambiguation (with pronominal disambiguation still to do).

Finally, knowledge of both the post's author as well as its dialog act classification could be used to detect and separate the multiple conversation threads within a session in the privacy-masked corpus. These experiments, however, would first require the investigator to identify and separate the threads for reference, which could be time-consuming.

5. Expand Privacy-Masked Chat Corpus

Our final recommendation for future work in this area is to increase the size of the privacy-masked corpus using the bootstrapping process described in Chapter III, Section A.5. The more data we have from the chat domain, the better any stochastic NLP technique used in a chat application should work. As noted in Chapter III, we highly recommend multiple annotators participate during the hand-verification step, using an established framework to guide them in their annotation decisions, whether they involve part-of-speech tagging, dialog act classification, syntax parsing, etc. Multiple annotators

serve two functions. First, they help provide a better corpus, since simple annotation mistakes can be caught through multiple eyes watching the process. More importantly, though having multiple annotators permits one to establish the inter-annotator agreement along with the associated Kappa statistic, which normalizes agreement to account for chance. Inter-annotator agreement can then be used to establish the "gold standard," or upper bound best possible performance, for a particular machine-learning method.

APPENDIX A: ACRONYMS

C2	Command and control
CMC	Computer-mediated communication
HMM	Hidden Markov Model
IM	Instant messaging
LDC	Linguistic Data Consortium
PCFG	Probabilistic context-free grammar
POS	Part-of-speech
MLE	Maximum likelihood estimate
NLP	Natural language processing
WSJ	Wall Street Journal
XML	Extensible Markup Language

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B: CHAT CONTRACTIONS

Word	Count	Alternate Tokenization	Word	Count	Alternate Tokenization
alot	7	a lot	itz	1	it z
alotta	1	a lot of	ive	1	I ve
arent	2	are nt	lotsa	1	lots a
couldnt	3	could nt	lotta	1	lott a
didnt	28	did nt	offa	1	off a
dint	1	di nt	shes	4	she s
doesnt	5	does nt	shouldnt	1	should nt
donno	3	don no	shouldve	1	should ve
dont	77	do nt	thats	45	that s
dontcha	1	do nt cha	tryina	1	tryin a
dunno	7	dun no	ur	21	u r
hafta	2	haf ta	wana	8	wan a
havent	3	have nt	whatcha	2	what cha
hes	4	he s	whats	41	what s
hows	8	how s	whys	1	why s
howz	2	how z	wonna	1	wonn a
ill	9	i ll	wouldnt	5	would nt
im	149	i m	wuts	1	wut s
ima	8	i m a	yall	12	y all
imma	3	i mm a	youre	3	you re
isnt	3	is nt	youve	1	you ve
its	69	it s			

Table 30. Contractions Encountered in Privacy-Masked Chat Corpus

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: CHAT EMOTICONS

Emoticons found in the privacy masked chat corpus are shown in Table 31. Most are apparent, although two classes bear specific mention. The first, indicated by three or more open/closed parenthesis/brackets such as ")))))", signify one half of a "hug". Thus, the following indicates 10-19-40sUser111 is being given a hug—(((((10-19-40sUser111))))). The second, indicated by ":*word*:", signify a command to the chat room system to display one of its built-in emoticons. Thus, the following indicates displaying a graphical emoticon showing a smiley face drinking a beer—":beer:".

(o Y o)	)))))))	:/	=(
(((	)))))))))	:@	=)
((((	)))))))))	:D	=A:A=-\
(((((	)))))))))	:O	=/
((((((	)))))))))))	:P	=D
(((((((	)))))))))))	:]	=O
((((((((	)))))))))))	:beer:	=[
(((((((((	)))))))))))	:blush:	=]
((((((((((	)))))))))))))	:love:	=p
(((((((((((	)))))))))))))))	:o *	>:->
((((((((((((	)))))))))))))))))	:p	@$$
(((((((((((((	)))))))))))))))))))	:tongue:	[[[[[[[[[[[[[[[[[[
((((((((((((((	)))))))))))))))))))))	:|	]:)
(((((((((((((((	)))))))))))))))))))))))))))	;)	]]]]]]]]]]]]]]]]]]]]
((((((((((((((((	)))))))))))))))))))))))))))))	;-(	^_^
(((((((((((((((((	)?	;-)	_
((((((((((((((((((	+*+*+*+*	;0	o.0
(((((((((((((((((((	-(	;]	o.O
((((((((((((((((((((	-_-	<3	o.o
(((((((((((((((((((((	=-s	<3's	o0
(((((((((((((((((((((	3333333	<33	o0o
(((((((((((((((((((((	:(	<333	o<|=D
(((((..	:)	<3333	oO
(__I__)	:-(	<33333	oOo
)))	:-)	<333333333	o_0
))))	:-@	<3333333333333333	o_O
)))))	:-o	<33333333333333333	xD

Table 31. Chat Emoticons Encountered in Privacy-Masked Chat Corpus

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D: CHAT ABBREVIATIONS

Abbreviation	Definition	Abbreviation	Definition
afk	away from keyboard	ltnc	long time no chat
bbl	be back later	ltns	long time no see
bbs	be back soon	ltnsea	ltns phonetic
brb	be right back	ltr	later
brbbb	brb variant	nm	not much
btw	by the way	omg	oh my god
cya	see you	omggg	omg variant
gm	good morning	rofl	rolling on floor laughing
gn	good night	rotflmao	rolling on the floor laughing my ass off
gtg	got to go	t/c	take care
j/k	just kidding	t/y	thank you
j/p	just playing	tc	take care
jk	just kidding	tdr	turbo diesel register
jw	just wondering	ty	thank you
lawl	laughing out loud (phonetic)	tyvm	thank you very much
lmao	laughing my ass off	w/b	welcome back
lmaoo	lmao variant	wb	welcome back
lmaooo	lmao variant	wc	who cares
lmaoooo	lmao variant	wth	what the hell
lmaooooo	lmao variant	wtf	what the f**k
lmfao	laughing my f**king ass off	y/w	your welcome
lol	laughing out loud	yvw	you very welcome
lolol	lol variant	yw	your welcome
lolololll	lol variant	yw's	yw variant
lool	lol variant		

Table 32. Chat Abbreviations Encountered in Privacy-Masked Chat Corpus

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] S. C. Herring, *Computer-Mediated Communication: Linguistic, Social, and Cross-Cultural Perspectives.* Amsterdam: John Benjamins, 1996.

[2] B. A. Eovito, "An assessment of joint chat requirements from current usage patterns," M.S. thesis, Naval Postgraduate School, Monterey, CA, U.S.A., 2006.

[3] E. Ivanovic, "Dialogue act tagging for instant messaging chat sessions," in *Proceedings of the ACL Student Research Workshop,* 2005, pp. 79–84.

[4] Linguistic Data Consortium, "About the Linguistic Data Consortium," August 2007, Available at http://www.ldc.upenn.edu/About/.

[5] M. Zitzen and D. Stein, "Chat and conversation: a case of transmedial stability?" *Linguistics,* vol. 42, no. 5, pp. 983-1021, 2004.

[6] M. Nystand, "The role of context in written communication," in *Comprehending Oral and Written Language* R. Horowitz and S. J. Samuels, Eds. San Diego: Academic Press, 1987.

[7] H. Sacks, E. A. Schegloff and G. Jefferson, "A simplest systematics for the organization of turn-taking for conversation," *Language,* vol. 50, no. 4, pp. 696-735, 1974.

[8] A. C. Garcia and J. Baker Jacobs, "The eyes of the beholder: understanding the turn-taking system in quasi-synchronous computer-mediated communication," *Research on Language and Social Interaction,* vol. 32, no. 4, pp. 337-367, 1999.

[9] M. R. Freiermuth, "Features of electronic synchronous communication: a comparative analysis of online chat, spoken and written texts," Ph.D. dissertation, Oklahoma State University, Stillwater, OK, U.S.A., 2002.

[10] W. Chafe and J. Danielewicz, "Properties of spoken and written language," in *Comprehending Oral and Written Language* R. Horowitz and S. J. Samuels, Eds. San Diego: Academic Press, 1987, pp. 83-113.

[11] J. Lin, "Automatic author profiling of online chat logs," M.S. thesis, Naval Postgraduate School, Monterey, CA, U.S.A., 2007.

[12] W. N. Francis, H. Kučera and A. W. Mackie, *Frequency Analysis of English Usage: Lexicon and Grammar.* Boston: Houghton Mifflin, 1982.

[13] C. D. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing.* Cambridge, MA: MIT Press, 1999.

[14] D. Jurafsky and J. H. Martin, *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Upper Saddle River, NJ: Prentice Hall, 2000.

[15] E. Brill, “Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging,” *Computational Linguistics,* vol. 21, no. 4, pp. 543-565, 1995.

[16] J. L. Austin, *How to do Things with Words.* Oxford: Clarendon Press, 1962.

[17] A. Stolcke, K. Ries, N. Coccaro, E. Shriberg, R. Bates, D. Jurafsky, P. Taylor, R. Martin, C. Van Ess-Dykema and M. Meteer, "Dialogue act modeling for automatic tagging and recognition of conversational speech," *Computational Linguistics,* vol. 26, no. 3, pp. 339-373, 2000.

[18] T. Wu, F. M. Khan, T. A. Fisher, L. A. Shuler and W. M. Pottenger, "Posting act tagging using transformation-based learning," presented at *The Proceedings of the Workshop on Foundations of Data Mining and Discovery, IEEE International Conference on Data Mining,* December 2002.

[19] F. Lundh, “Python ElementTree module,” August 2007, Available at http://effbot.org/zone/element-index.htm.

[20] M. P. Marcus, B. Santorini, Marcinkiewicz M. and Taylor A., "Treebank-3," 1999, Available at http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC99T42.

[21] S. Bird, E. Klein and E. Loper, "NLTK: the Natural Language Toolkit," ver. 0.7.4, August 2007, Available at http://nltk.sourceforge.net/index.php/Main_Page.

[22] S. Bird, E. Klein and E. Loper, *Natural Language Processing in Python.* August 2007, Available at http://nltk.sourceforge.net/index.php/Book.

[23] T. M. Mitchell, *Machine Learning.* New York: McGraw-Hill, 1997.

[24] G. F. Luger, *Artificial Intelligence : Structures and Strategies for Complex Problem Solving.* ,4th ed., New York: Pearson Education, 2002.

[25] N. Schemenauer, "Back-propagation neural network," March 2007, Available at http://arctrix.com/nas/python/bpnn.py.

[26] E. Ivanovic, "Automatic utterance segmentation in instant messaging dialogue," in *Proceedings of the Australasian Language Technology Workshop,* 2005, pp. 241-249.

[27] R. Hwa, "Supervised grammar induction using training data with limited constituent information," in *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics,* 1999, pp. 73-79.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. LorRaine T Duffy, PhD
   SSC San Diego Code 246207
   San Diego, California